MASTER EN SCIENCES INFORMATIQUES

Contributions à un système de dialogue homme-machine à composante orale

Brousmiche, Jean-Pierre

Award date:
1987

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grangagnage , 21
B - 5000 Namur

Contributions à un système de dialogue homme-machine a composante orale

Mémoire présenté par

Jean-Pierre Brousmiche

en vue de l'obtention

du titre de licencié et maître en informatique.

Année académique 1986 - 1987

Remerciements.

Cette année de travail m'a apporté de nombreuses connaissances dans le domaine de la compréhension de la parole par la machine. Cela m'a beaucoup intéressé et je voudrais pour cela remercier un certain nombre de personnes sans lesquelles je n'aurais pas pu réaliser ce mémoire.

En premier lieu, je remercie le père J. Berleur pour avoir accepté d'être le promoteur de ce mémoire ainsi que pour avoir consacré une partie de son temps à lire cet ouvrage et me conseiller quant à sa rédaction.

Ensuite, mes remerciements s'adresseront particulièrement à Monsieur P. Mousel avec lequel j'ai travaillé et qui m'a guidé tout au long de l'année au travers de ce domaine de recherche sans cesse en évolution.

Enfin, je n'oublierai pas de remercier Messieurs G. Deville, M. Lorant et S. Simonet qui m'ont très gentiment fourni des informations et toute documentation bien utile pour comprendre les multiples facettes de la compréhension de la parole par la machine, ainsi que Madame A. Peremans-Degbomont pour l'aide à apportée à la frappe de ce mémoire.

Chapitre 1.

Introduction à la communication entre l'homme et la machine.

Avant toute chose, il s'agit de planter le décor, de définir le cadre général.

Dans ce premier chapitre, je commencerai par parler brièvement de la communication, des moyens de communiquer et de la différence entre le langage et la parole. Ensuite, j'aborderai la notion de dialogue et cela en particulier dans la communication entre l'homme et la machine. On sera ensuite tout naturellement amené à aborder des problèmes un peu plus techniques en parlant des différents types de langages utilisés pour dialoguer avec ces machines.

Commençons donc par nous poser la question de savoir ce qu'est la communication. Bien sûr, il n'est pas dans notre propos de faire toute une dissertation sur la communication mais il est important de tenter d'en dégager certaines caractéristiques et de voir dans quelles mesures celles-ci pourraient être utiles dans le cadre de la communication homme-machine.

1.1 la communication.

En réfléchissant un peu, on peut affirmer que proner un monde sans communication, c'est prêcher contre toute société constituée. La communication est la glu qui tient ensemble toute société, humaine ou animale. Et les hommes ont besoin de vivre en société. La communication est donc un besoin vital pour elle. Cette communication permet de créer des liens, de transmettre et de recevoir des informations.

La communication est donc chose de première importance dans ce monde. Ainsi, on peut dire qu'il y a communication dès que l'on parvient à transmettre un message à un interlocuteur. Il est bien sûr préférable que ce message soit compris par le récepteur. Aussi entre l'émetteur et le récepteur un code commun pour la compréhension du message doit exister.

Cette communication peut se faire d'une infinité de façons et de nombreux moyens de communication existent. Mais il en est un

possédant une situation privilégiée au sein de cette multitude c'est-à-dire la parole. Celle-ci peut d'ailleurs être considérée comme une spécificité et un déterminant de la réalité humaine.

Voyons donc ce qu'est la parole et faisons donc une première distinction entre ce que recouvrent les mots "parole" et "langage".

L'homme exprime généralement ses idées,ses sentiments, ses volontés, et ses sensations par la parole. Mais la parole n'est qu'un des moyens naturels de communication pour l'homme. C'est ce qu'on appelle le langage verbal. Néanmoins, tout un langage non-verbal existe également. Ce dernier étant en fait tout aussi courant que le langage verbal. Ce langage non-verbal est par exemple, l'écriture, les mouvements et les positions du corps, le regard et les expressions du visage, les bruits et même les sensations physiologiques telles que rougir ou pâlir. Il est possible de les grouper en différentes classes qui sont les moyens auditifs, visuels,tactiles et olfactifs.
Le mot "langage" recouvre ces deux classes qui sont le langage verbal et le langage non-verbal.

Considérons alors les caractéristiques principales de la parole qui la rapprochent ou la séparent des autres moyens de communication.

Parmi les points communs de la parole avec les autres moyens de communication, on trouve :

- La fonction de communiquer un message.
- L'utilisation de signes arbitraires plutôt que de symboles.

En effet, beaucoup de communications utilisent également des signes arbitraires. On peut citer, par exemple, le code de la route où un cercle désigne une autorisation ou une interdiction, un triangle un danger et un rectangle une indication. Néanmoins, pour ce qui est de la parole, il faut noter que celle-ci possède également divers éléments symboliques, en particulier pour ce qui est de l'ironie.

La spécificité de la parole par rapport aux autres moyens de communication se situe selon A. Martinet { Martinet 56 } au niveau

de la construction des langages humains car ceux-ci sont basés sur une double articulation.
La première articulation spécifie, dans la terminologie d'A. Martinet est que nos langages se décomposent en chaines vocales qui sont intrinsèquement non-significatives et qu'on nomme les phonèmes ou sons du langage. Ces unités minimales ont une forme mais pas de signifié. La seconde articulation est celle selon laquelle nos langages peuvent également se décomposer en un certain nombre d'unités minimales possédant à la fois une forme et un sens. Celles-ci sont appelées les monèmes ou mots. C'est grâce au double codage réalisé tout d'abord sur les phonèmes (pour obtenir des monèmes) puis ensuite sur ces monèmes que que nous pouvons exprimer l'infinité des situations pouvant survenir devant nous.

La puissance du langage humain est donc de pouvoir tout exprimer à partir d'une cinquantaine de ces unités minimales: les phonèmes et uniquement par combinaison de ceux-ci. C'est la réalisation de ce codage qui détermine une langue et qui permet de différencier le français par exemple de toutes les autres langues (plus de 56 familles recencées à ce jour).

1.2 Le dialogue.

Après avoir abordé les notions de communication et de langage, on en arrive à évoquer celle du dialogue. Ici encore, il est hors de question de réaliser une étude complète du dialogue mais plutôt de dégager un ensemble de propriétés et de fonctionnalités utiles pour la compréhension du dialogue entre l'homme et la machine.

Le premier fait à remarquer est que, sans dialogue, la communication se réduit souvent à une simple transmission d'informations non-satisfaisante pour l'homme. On est toujours ennuyé si l'auditeur reste sans réaction et ne répond pas, que celui-ci soit homme ou machine.
Cela montre qu'un système complet basé pour la communication avec l'extérieur sur la reconnaissance ou la compréhension de la parole se doit de posséder un module de dialogue.

Les fonctions de ce dialogue sont alors :

- Etablir un contact entre les interlocuteurs. Cela leur permet de fixer l'objet de leur discussion et de déterminer leurs positions réciproques.

- De permettre, bien évidemment, un échange d'informations entre les différents interlocuteurs.

- De confirmer ou de valider des éléments du dialogue. Cela correspond à un besoin psychologique essentiel car on a toujours besoin d'être certain qu'on nous écoute et de s'assurer qu'on nous a compris.

- D'orienter l'échange d'informations et de modifier les sujets de dialogue pour obtenir l'information pertinente et nécessaire. Celui qui va orienter le dialogue sera considéré comme le maitre de l'échange, même si, dans un dialogue authentique, il n'y a ni "maître", ni "esclave".

Le dialogue dans la communication homme-machine.

La parole étant le moyen de communiquer de l'information le plus naturel à l'homme, il est logique de penser à créer un interface de dialogue entre cet homme et cette machine basé sur la parole. Néanmoins, si on considère le dialogue dans la vie courante, on s'aperçoit aisément que celui-ci fait intervenir tout ce qu'on appelle le non-verbal c'est-à-dire les gestes , les attitudes, les mimiques ...
Une des grandes difficultés rencontrées alors lors de la création de procédures de dialogue en communication homme-machine est d'aboutir à une compréhension tout en se limitant à l'échange d'informations par le canal verbal.

Beaucoup de travaux ont déjà été réalisés et d'appareils construits sur base d'un " interface oral " et ils ont mis en évidence toute une série d'avantages et de désavantages de cet interface dans différents domaines d'application. Il est une fois

de plus impossible de les énumérer un à un et de discuter chacun de ces cas mais il est possible, par curiosité, de signaler quelques réalisations d'applications de cet interface (Pierrel 81, Lorant 86, ...).

- La saisie de données : dans bien des cas, les claviers ou encodeurs sont avantageusement remplacés par un mode de saisie oral.

- L'identification de personnes pour des applications nécessitant un accès privilégié et une grande sécurité.

- L'enseignement assisté par ordinateur.

- La saisie des commandes, que ce soit pour l'aide aux handicapés qui ne peuvent utiliser les moyens traditionnels de commandes (leviers, boutons, claviers ...) ou pour permettre une plus grande précision dans le maniement d'appareils par des personnes valides. Par exemple, le cas du chirurgien devant réaliser les réglages du microscope au pied car il a déjà les deux mains occupées. Une commande vocale du microscope serait plus précise car le pied n'est pas adapté à un travail de précision.

- La bureautique : réalisation d'agendas vocaux, ...

- L'interrogation de bases de données ou de centres de renseignements par des personnes non habituées au maniement des machines.

etc ...

Il est possible de rassembler les avantages de l'interface vocal entre l'homme et la machine en disant que :

- La communication entre l'homme et la machine s'opère par le moyen le plus naturel à l'homme et celui qu'il a le plus tendance à utiliser : la parole.

- La parole est le moyen de communication le plus rapide que

l'homme possède pour transmettre des informations.

- L'accès aux machines est facilité pour ceux qui pour diverses raisons, ne peuvent que difficilement y avoir accès.

- La parole permet la mobilité et la réalisation d'autres tâches.

et bien d'autres encore laissés à l'imagination du lecteur.

Cependant, il y a toujours des inconvénients qu'il importe de mesurer et qui peuvent s'opposer à la réalisation de systèmes de dialogue entre l'homme et la machine par voie orale.

Ainsi, se demande Weizenbaum { Weiz 84 }, n'y a-t-il pas des objectifs incompatibles avec les machines ? Même si toutes les informations nécessaires à la compréhension de la parole peuvent être suffisamment modélisées pour réaliser un système de compréhension de la parole complet et pouvant fonctionner avec le langage de tous les jours, ne faut-il pas s'interroger sur les objectifs poursuivis par ces recherches et examiner les préjudices éventuels qui pourraient en résulter, tels par exemple la généralisation des écoutes téléphoniques.

Associé à ce premier problème, sur un plan plus technique, on peut en ajouter un second qui est de se demander si il est possible d'obtenir une bonne compréhension de la parole par l'ordinateur étant donné le nombre de difficultés à résoudre. On peut citer par exemple les problèmes d'analyse acoustique et phonétique du signal. Toutes les personnes s'exprimant de façon différente, avec des accents et des rythmes de prononciation différents ; le fond sonore qui se superpose à l'énoncé de la phrase et qui peut rendre inaudible ou incompréhensible la compréhension de celui-ci, la machine ne pouvant pas discerner ce qui est la voix de son interlocuteur de ce qui est bruit parasite car elle ne s'appuie que sur le langage verbal ; des problèmes de limites causés par les informations incomplètes en matière de connaissance linguistique ; ...

On peut en arriver à la conclusion que tout ce qui entre en compte dans la compréhension de la parole ne peut être formalisé et qu'il existe en fait des objectifs propres aux machines et d'autres propres à l'homme. Si la parole se trouve être le propre de l'homme, il y aurait une limite intrinsèque au traitement automatique de la parole par l'ordinateur.

Dans le but de pouvoir maitriser les nombreux paramètres liés à la compréhension de la parole par la machine, les chercheurs ont développé différents langages d'interaction avec la machine. Ces langages obéissent à un ensemble de contraintes portant sur le vocabulaire utilisé, sur la façon de s'exprimer face à la machine, sur la syntaxe à utiliser, ... Ces différents langages étant de plus en plus évolués avec le temps en ce sens qu'ils possédent de moins en moins de contraintes.

Venons-en donc à considérer ces différents langages verbaux que l'homme a créés pour communiquer avec la machine. On peut classer ces différents langages par type { Lorant 86 }:

1.3 Typologie des langages et de la communication homme-machine.

La compréhension par la machine de la langue naturelle parlée par une personne de la rue étant extrêmement difficile et nécessitant énormément de connaissances, on a développé progressivement divers langages, sous-ensembles de la langue naturelle, nécessitant des connaissances moins importantes et des traitements moins complexes pour réaliser cet objectif de compréhension de la parole.

Dans la littérature, on découvre principalement 4 types de langages. Ces langages peuvent par ailleurs être plus pratiques que la langue naturelle dans le cadre de certaines applications.

1.3.1 Les mots isolés et les séquences de mots isolés.

Un premier type de langage consiste à émettre des messages en ne se servant que de mots choisis dans un vocabulaire très restreint. Chacun de ces mots est séparé du précédent et du suivant par une pause.

Ce type de langage peut servir dans des applications telles que la commande d'un robot devant effectuer un mouvement, tel par exemple, soulever une éprouvette. Le langage utilisé donnerait ceci :

" soulever (pause) éprouvette "

Ces langages sont relativement pauvres mais conviennent très bien au traitement automatique de la parole. Ce genre de langage est d'ailleurs essentiellemnt utilisé pour la commande vocale de machines.

1.3.2 Les langages artificiels.

Ce deuxième type de langage réalise une évolution par rapport aux mots isolés en ce sens qu'il permet une continuité dans l'élocution. Il n'est plus nécessaire de s'arrêter un instant entre chaque mot. Les contraintes proviennent alors de la syntaxe à utiliser pour construire des phrases. Généralement, cette syntaxe est assez pauvre et le locuteur ne peut se permettre aucun écart par rapport à celle-ci.

Ce type de langage peut servir dans le cas de commandes de processus { Pierrel 81 } mais nécessite pour son utilisation toute une phase d'apprentissage du vocabulaire et de la syntaxe.

1.3.3 Les langages quasi (pseudo)-naturels.

Il y a pour ce troisième type de langage une réelle volonté de se rapprocher du langage que nous utilisons tous les jours. Pour cela, un minimum de contraintes ont été placées des points de vue lexical et syntaxique.

Néanmoins, il existe toujours des contraintes, celles-ci portant sur le domaine d'application. En effet, dans ce type de langage, une phrase énoncée par le locuteur ne peut être comprise que d'une seule façon, celle-ci dépendant du domaine d'application dans lequel on se situe.

Ces langages possèdent alors l'immense avantage de ne nécessiter aucune phase d'apprentissage pour leur utilisation. Il permet donc aux systèmes d'être accessibles par tout un chacun.

Cette façon de voir les choses oblige alors les concepteurs à considérer la parole avec tous ses problèmes d'altérations, de syntaxe et de vocabulaire utilisé.

1.3.4 Les langages naturels.

C'est le langage de tous les jours que chacun utilise. Il n'existe sur ces langages aucune contrainte autre que les contraintes de la grammaire et du vocabulaire qui existent dans toutes les langues . Il n'y a même pas de contrainte sur les sujets de conversation qui peuvent être abordés.

1.4 Conclusion.

La parole constitue donc un moyen privilégié et de grand intérêt pour la communication avec la machine. Elle possède bon nombre d'avantages mais pose d'immenses problèmes à résoudre. Des études ont été réalisées et des systèmes développés pour tenter de trouver des solutions à tous ces problèmes. Ces systèmes utilisent des langages plus ou moins évolués, avec plus ou moins de contraintes de vocabulaire, de syntaxe, de façon de s'exprimer ... pour pouvoir solutionner les problèmes un à un.

Mais bien que certains langages soient plus contraignants que d'autres, ils sont parfois bien adaptés à certaines applications qui ne nécessitent pas un système très complexe de traitement de la langue naturelle pour être mises en oeuvre. Par exemple, pour

commander des processus de production ou des robots, un langage de mots isolés ou artificiel est très précis et très pratique. Mais il possède l'inconvénient de demander un apprentissage. Par contre, une application destinée au grand public doit pour pouvoir être réalisée en langage naturel et n'exiger aucun apprentissage. Il faut donc choisir le langage que l'on va utiliser en fonction du type d'application pour lequel il est destiné. Dans le cadre de ce travail, nous nous intéresserons spécialement au traitement des langages pseudo-naturels.

Chapitre 2.

Les principes généraux des systèmes de reconnaissance de la parole.

Comme son titre l'indique, ce chapitre sera consacré aux principes généraux des systèmes de reconnaissance et de compréhension de la parole continue. Je commencerai donc par faire la distinction entre la simple reconnaissance et la compréhension de la parole. Ce sera plus particulièrement cette dernière qui nous intéressera dans la suite du travail. J'aborderai ensuite les notions linguistiques et acoustiques de la parole, utilisées habituellement en compréhension de la parole. Et de là, je serai amené à différencier plusieurs niveaux possibles d'analyse de la parole, lesquels seront alors expliqués.

Ensuite, après une petite conclusion, je présenterai les divers principes de base sur lesquels se développent les systèmes de compréhension de la parole continue. Je parlerai des différents composants d'analyse habituellement utilisés, des diverses interactions entre ces composants et des stratégies de contrôle possibles à l'intérieur de ces systèmes.

2.1 Reconnaissance et compréhension de la parole.

On fait généralement la distinction entre deux types de systèmes utilisant la parole comme interface. On distingue les systèmes de reconnaissance de la parole (Speech Recognising System) et les systèmes de compréhension de la parole (Speech Understanding System). Mais on constate que la plupart des travaux actuels sont en fait tournés vers le deuxième type.

S.R.S. : Dans les systèmes de reconnaissance de la parole, on ne porte pas son attention sur la signification du message. On essaie seulement de retrouver les unités de base (phonèmes, mots...) présentes dans le message. On n'emploie pas pour cette tâche d'informations sémantiques (sur la signification des

mots) ou pragmatiques (tenant compte du contexte et de l'environnement).

S.U.S. : Dans ce cas, l'objectif est de transformer le message en une représentation particulière à laquelle on attribue un sens. Le fait de comprendre la signification du message permettra par la suite de déclencher une certaine action en réponse.

Lorsqu'on envisage de créer un système de compréhension de la parole, on peut considérer celle-ci sous divers aspects. Premièrement sous un aspect acoustique et ensuite sous un aspect linguistique.

2.2 Aspect acoustique et aspect linguistique de la parole.

Considérer la parole sous son aspect acoustique, c'est la considérer sous la forme du signal physique obtenu à la sortie du microphone. Ce signal physique possède évidemment un ensemble de caractéristiques. Citons les principales:

- Il contient un nombre d'informations très élevé (jusqu'à 100000 bps) { Haton 85 }.
- Il apparaît comme un semi-continuum dans lequel il est très difficile de déterminer les frontières entre les mots. Pour réaliser une segmentation du message entre les mots, il faut utiliser des procédures très complexes { Haton 85 }.
- Il dépend fortement du locuteur. Deux personnes répétant une même phrase la prononceront de façon différente, à des vitesses différentes, avec des intonations différentes.
De plus, une même personne est incapable de répéter deux fois la même phrase et d'obtenir deux fois le même signal à la sortie de microphone.
- Il dépend des bruits environnant le locuteur. En effet, lorsqu'on regarde le signal produit à la sortie d'un microphone, il est absolument imposible de discerner à coup sûr les parties correspondant à la voix des parties provenant des bruits parasites.

Ces deux parties se combinant continuellement.

La figure II.1 représente un exemple de signal obtenu à la sortie du microphone.

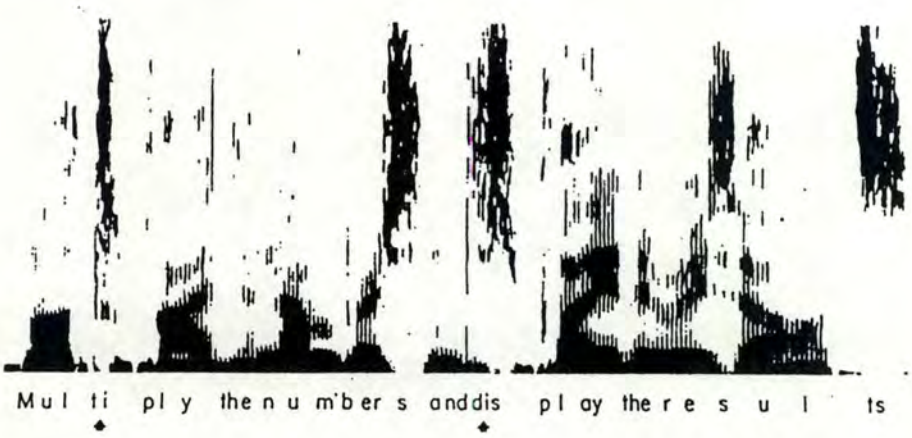


Figure II.1. : Exemple de signal reçu à la sortie du microphone. Tiré de (Lea 80).

Le second aspect du signal (en fait de la parole prononcée) est son aspect linguistique c'est-à-dire l'aspect grammatical de la parole et du langage et son interprétation du point de vue de son sens. L'interprétation linguistique du signal constitue une opération très complexe. Cela est dû, en premier lieu, aux caractéristiques intrinsèques de ce signal mais également au fait que les processus de perception et de compréhension de la parole reposent sur l'interaction de plusieurs niveaux d'analyse linguistique. On a, à ce jour, pas encore bien compris et expliqué tous ces niveaux d'analyse.

Sans entrer dans les détails des diverses analyses acoustiques et linguistiques du signal, on peut donner un aperçu des quelques principes de base qui peuvent servir de guide lors du traitement automatique de la parole.

2.3 Les différentes analyses.

Pour commencer, présentons un modèle hiérarchique de la perception humaine qui est proche de celui de Libermann { Liber 70 } (Voir figure II.2). Ce modèle nous montre toutes les transformations que subit une idée qui doit être émise pour être traduite en signal acoustique.

L'interprétation linguistique du signal acoustique s'appuie sur { Lorant 86 } :

" [1] l'utilisation de connaissances multiples et variées :

+ acoustique : pour le traitement du signal.
+ phonétique : décrivant les caractéristiques des sons de la langue.
+ phonologiques : décrivant les phénomènes d'altération de ces sons dans des contextes donnés.
+ prosodiques : rendant compte des rythmes, des intonations et mélodies de la voix.
+ lexicales : lié aux unités signifiantes.
+ syntaxiques : comprenant les règles de formation de phrases correctes.
+ sémantiques : prenant en compte les problèmes de la signification des énoncés.
+ pragmatiques : décrivant l'univers dans lequel on se place (plus spécifiquement dans le cadre de dialogues).

[2] sur différents traitements et leurs interactions. "

Classiquement, on observe quatre niveaux d'analyse qui sont les niveaux phonétique, lexical, syntaxique et sémantique.

Dans le cadre notre système de compréhension de la parole, nous respecterons cette découpe en niveaux d'analyse. Il est donc important de savoir ce que réalisent ces différentes analyses. Commençons alors par celle la plus proche du signal, à savoir :

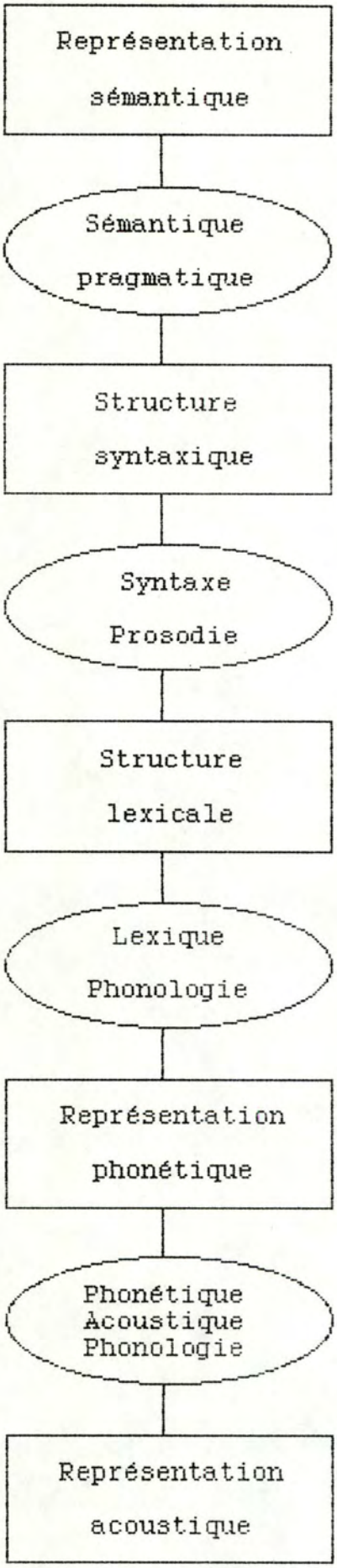


Figure II.2 : Modèle hiérarchique de la perception humaine.

2.3.1 L'analyse acoustico-phonétique. { Lea 80 , Carb 85 }

Ce premier niveau d'analyse a pour but de retrouver la structure phonétique de la phrase. Cette structure phonétique se présente comme un découpage du signal en unités minimales de sons identifiées et reconnues sur celui-ci. Ces unités minimales de sons peuvent correspondre à diverses choses. On peut considérer que ce sont des allophones [un allophone étant un ensemble d'éléments minimaux de son d'une langue donnée ayant les mêmes caractéristiques de structure], des phonèmes, des diphones, des syllabes ou encore des combinaisons de ceux-ci. Cette analyse se fait en deux temps. Après une opération de segmentation du signal, il y a une phase d'identification de ces segments car il ne faut pas seulement pouvoir les situer mais également pouvoir les identifier. Malheureusement, ces deux opérations sont rendues complexes par le caractère continu du signal et également par le fait de phénomènes physiques liés au phénomène de production de la parole. On peut citer en exemple la mauvaise articulation des locuteurs, le bruit de fond, ... Cela induit alors des phénomènes de chevauchement et de déformation des unités minimales de parole. Les unités minimales peuvent alors ne plus ressembler du tout à leur définition théorique et il est pour cette raison, difficile de les reconnaître.
En fait, il n'y a pas de solution générale de ces problèmes { Léa 80 }. En général, cette analyse ne produit pas une solution unique. Le résultat se présente souvent sous la forme d'un *treilliss phonétique* qui combine les différentes solutions retenues. Il faudra donc que les autres modules utilisant les résultats fournis par cette analyse phonétique soient conscients que ceux-ci possèdent un taux non-négligeable d'erreurs. Ces autres modules pourront par la suite, à l'aide d'autres critères et d'autres connaissances, déterminer la solution la plus probable à défaut de trouver la solution exacte.

2.3.2 L'analyse lexicale.

Ce type d'analyse a pour but de tenter de reconstruire, à partir du treillis d'unités phonétiques fourni par l'analyse

phonétique, la structure lexicale de l'énoncé. Il va, en d'autres mots, tenter de retrouver les mots présents dans l'énoncé à partir des unités de base révélées par l'analyse phonétique.
Il faut pour cela que ces mots appartiennent au vocabulaire accepté par le système. Ici non plus, les résultats obtenus ne sont pas exempts d'erreurs car il ne faut pas oublier la quantité de problèmes présents causés par

- l'absence de frontière entre les mots dans les représentations acoustique et phonétique.
- la dépendance du signal vis à vis des bruits extérieurs au message.
- les erreurs qui ont pu être introduites dans la représentation phonétique fournie par le niveau inférieur.

etc ...

2.3.3 L'analyse syntaxique.

A partir des hypothèses de mots réalisées au niveau lexical, l'analyse syntaxique essaie de reconstituer la structure syntaxique de l'énoncé prononcé par le locuteur. Cette structure syntaxique obéira bien entendu aux règles de la grammaire utilisée dans le système.

Beaucoup de grammaires relativement générales ont été développées pour le traitement du langage naturel par ordinateur. Malheureusement, elles ne travaillent principalement que sur des sous-ensembles des langues naturelles et dans le cadre de tâches assez restreintes.

Pour cette raison, les différents constructeurs adaptent eux-mêmes les grammaires à leur tâche sur base de connaissances souvent empiriques et informelles.

Les représentations syntaxiques obtenues sont bien évidemment incertaines et même parfois incomplètes et il faudra de nouveau en tenir compte lors de l'exploitation de celles-ci par les autres niveaux.

Le chapitre 3 sera consacré à l'étude des grammaires utilisées pour les analyses syntaxique et sémantique.

2.3.4 L'analyse sémantique.

A ce niveau d'analyse, on veut essayer de comprendre la signification d'un énoncé, ce qui est le but final de tout système de compréhension de la parole. Ce niveau a pour tâche de fournir une interprétation sémantique adéquate de chaque phrase de l'énoncé dans le système de dialogue.

Cette interprétation dépend

- de la signification des mots qu'elle contient.
- des relations entre ces mots.
- du contexte d'énonciation dans lequel elle est produite, dans le cadre plus particulier d'un dialogue et d'une application donnée.

Pour réaliser cette interprétation sémantique, on se base habituellement sur la structure syntaxique et/ou lexicale déjà construite de la phrase.

Beaucoup de problèmes apparaissent encore lorsqu'il faut réaliser cette analyse en raison de l'ambiguïté naturelle du langage, (polysémies, homonymes, importance du contexte d'énonciation) et de l'ambiguïté provenant du processus de reconnaissance (erreurs, homophones…).

Diverses grammaires peuvent être utilisées pour construire une interprétation sémantique de l'énoncé. Celles-ci seront expliquées dans le chapitre suivant.

Cette interprétation sémantique va également se heurter au problème de l'adoption d'un formalisme adéquat de représentation. Elle va également devoir résoudre divers problèmes tels que la définition du sens des mots, le traitement des synonymes au niveau

des mots et aussi au niveau des syntagmes tels, par exemple, "ne pars pas" et "reste", les inférences de propositions diverses à partir d'un énoncé (implications logiques, déductions, ...) { Méloni 83 }.

2.3.5 Conclusion.

Lorsqu'on étudie de près les phénomènes acoustique et linguistique, on en arrive à la conclusion que les caractéristiques essentielles du problème de la compréhension automatique de la parole sont { Lorant 86 }:

" - l'absence de solutions générales aux problèmes qu'elle pose.
- le caractère non déterministe du processus, essentiellement dû à l'environnement incertain dans lequel il prend place (variabilité du signal, présence d'erreur à tous les niveaux)
- et finalement, en tant que conséquence des deux premières, la nécessité d'intégrer toutes les connaissances disponibles à tous les niveaux d'analyse, jusqu'à une collaboration étroite afin de réduire l'incertitude et l'explosion combinatoire de solutions qui en découle. Ainsi, comme on le verra, la connaissance utilisée à un niveau d'analyse peut également jouer comme une contrainte à un niveau plus bas. "

Les chercheurs dans ces domaines ont donc réalisé des approches progressives des problèmes intégrant de plus en plus de difficultés, par exemple, des langages de plus en plus évolués. Cela leur a permis de résoudre les problèmes un à un en leur apportant la solution la plus adéquate possible.

2.4 La compréhension de la parole continue { Lea 80 }.

Diverses approches de la compréhension de la parole sont possibles. Une différence est faite entre les approches qui ne font intervenir que des théories de types mathématiques et statistiques

pour reconnaître les différentes unités composant le signal , et d'autres qui, en plus, utilisent des connaissances linguistiques. Les approches mathématiques ne font que comparer le signal avec des définitions théoriques pré-enregistrées des mots du vocabulaire.

Les approches qui font intervenir des considérations linguistiques en relation avec la perception de la parole affirment, contrairement aux modèles purement mathématiques, que le signal acoustique et sa seule comparaison avec des modèles enregistrés n'est pas suffisant pour déterminer le message. D'autres sources de connaissance doivent être intégrées pour résoudre le problème de la reconnaissance et de la compréhension de la parole . Ces systèmes veulent développer quelque chose qui s'apparente fonctionnellement à l'homme. Ce sont ces approches qui nous intéresseront par la suite car ce sont les plus fréquentes et celles utilisées dans le système mis en oeuvre à Nancy. Ces systèmes désirant se développer sur cette base présenteront une certaine similarité avec le modèle hiérarchique de Libermann et font usage des différents types d'analyse que nous venons de mentionner.

Une des raisons principales de l'emploi de schémas de reconnaissance plus orientés vers la linguistique est qu'ils rendent plus aisée la compréhension dans un cadre très général. En effet, il a été prouvé que les approches orientées mathématique donnent de meilleurs résultats et dans un temps plus court lorsqu'on se situe dans un contexte très limité. Mais si on s'intéresse à des systèmes multi-tâches, les méthodes incluant des notions de linguistique prévalent sur les autres { Lea 80 }.

Les principales composantes de ces systèmes, en se référant au modèle de Libermann, sont :

- le décodage acoustico-phonétique
  = extraction de paramètres acoustiques, extraction et identification des unités de base composant le message.
- l'analyse lexicale
  = pour retrouver les mots présents dans l'énoncé.

- l'analyse sémantique et syntaxique
  = pour reconstituer la structure de la phrase d'un point de vue grammatical et voir si elle peut avoir un sens.
- l'analyse sémantico-pragmatique
  = pour comprendre la signification de la phrase en tenant compte de son contexte d'application.

Un cinquième niveau d'analyse, totalement indépendant des autres est:

- l'analyse prosodique
  = analyse de l'intonation, du rythme de la phrase

D'autres tâches accompagnent la détermination de ces principales composantes du système :

- la modélisation des connaissances utilisées c'est-à-dire la définition des connaissances que l'on va utiliser et le choix d'un formalisme de représentation et de mise en oeuvre.

- la définition d'une structure de contrôle (influençant fortement le choix du formalisme de représentation et de mise en oeuvre).

La suite de ce chapitre est consacrée à l'explicitation de ces différentes composantes, des stratégies de contrôle et de l'interaction entre les différents modules.

2.4.1 Les composantes d'un système de traitement de la parole.

2.4.1.1 Décodage acoustico-phonétique. (Lea 80).

Le signal acoustique est reconnu comme étant un phénomène complexe. Les sons utilisés par les locuteurs sont structurés de façon très différentes dans les différents langages. Une première

question à se poser est alors: " Quelles sont ces unités minimales de sons utilisés dans ce langage ? ". Un grand choix se présente habituellement. On y trouve les allophones, les phonèmes, les diphones, les syllabes… ou même certaines combinaisons de ceux-ci. Il est difficile d'en choisir une comme étant la meilleure pour le décodage acoustico-phonétique. Il est évident que chacune possède ses avantages et ses inconvénients.
Les plus importants de ceux-ci sont repris sur le tableau II.1.

Il semble cependant que l'unité la plus utilisée soit le phonème. Ce terme est, comme le signale { Lea 80 } " … utilisé pour l'ensemble complet des allophones [un allophone étant un ensemble d'éléments minimaux de son d'une langue donnée ayant les mêmes caractéristiques de structure] se comportant de manière similaire et ne se différenciant pas de manière significative les uns des autres au sein d'une même langue. ".

Une autre question à se poser est : " Peut-on dissocier ces unités minimales (les segments) au sein du signal et les identifier ? ". Les résultats obtenus par les experts pour la segmentation phonétique et l'identification des unités décelées sans faire appel à aucune connaissance syntaxique, sémantique ou à des contraintes sur le vocabulaire employé, sont de l'ordre de : 75 % de segments correctements repérés et reconnus, 15 % repérés mais mal identifiés et 10 % de segments non repérés. Le fait que des experts, travaillant depuis des années sur le sujet, réalisent encore un nombre non négligeable d'erreurs montre qu'il est assez illusoire qu'un composant automatique soit capable de segmenter et d'identifier ces segments avec certitude en utilisant uniquement des informations acoustiques { Woods 73 }.

Demandons-nous alors quelle est réellement la fonction de cette composante. Cette première étape d'analyse a pour fonction d'assurer :

1°) le découpage ou la segmentation du continuum constitué par le signal acoustique numérisé (obtenu à partir d'un énoncé de locuteur), en unités discrètes correspondant aux phonèmes successifs que le locuteur a voulu prononcer.
2°) identifier tous ces segments acoustiques qui ont été

détectés.

Il est vraiment difficile d'effectuer en même temps ces deux opérations. Aussi, en général, réalise-t-on d'abord la phase de segmentation et ensuite la phase d'identification. Ces deux opérations se heurtent malheureusement à différents problèmes qui entraînent un certain taux d'incertitude dans les résultats.

Les premiers problèmes qui apparaissent sont ceux que l'on découvre en tentant de réaliser la segmentation du signal en unités minimales. Déjà, des erreurs peuvent apparaître. Elles sont dues principalement à :

- l'absence de frontière manifeste entre les différents segments sur le signal.

- l'absence d'une représentation satisfaisante de la connaissance acoustico-phonétique.

Les différents segments peuvent se recouvrir et plusieurs segments peuvent être regroupés en un seul ou également un segment peut être décomposé en plusieurs autres. Cela peut être dû au fait que le locuteur parle trop vite ou trop lentement, qu'il articule mal, que la grammaire oblige à faire une liaison entre deux mots.... Une réelle incertitude dans les résultats apparaît à ce niveau.

Ensuite, d'autres problèmes apparaissent lors de l'identification de ces segments. Il faudrait d'abord qu'ils soient bien séparés. Et en plus, un même phonème prononcé se présente toujours de façon différente sur le signal à la sortie du microphone. Il existe toujours un écart entre la forme du phonème sur le signal et la forme du phonème établi théoriquement. On n'obtiendra donc jamais une structure phonétique unique mais plus généralement un ensemble de solutions appelé *treillis phonétique*. Ce sont alors les autres niveaux qui utilisent ce treillis phonétique et qui vont tenter de supprimer les ambiguïtés présentes en utilisant d'autres sources de connaissance.

Phonological Unit	Possible Advantages	Possible Disadvantages
Allophone	1. Certain ones are easily identifiable acoustically.	1. Instrumentation is not yet sophisticated enough for the task.
	2. Some word boundaries are indicated by allophones.	2. The total number of allophones can be excessively large.
	3. They reduce the need for rules at a lower level.	3. Many allophones are very dependent on their environment.
Phoneme	1. The number of distinctive phonological classes is small.	1. Phonemes are not easily determined acoustically.
	2. Phonemes map the most directly to lexicon entries in present computer dictionaries.	2. Some sounds belong equally well to more than one phoneme.
		3. Many rules are needed at both lower and higher linguistic levels.
Diphone	1. Transitional information is included.	1. The total number of diphones can be relatively large.
	2. Some coarticulation rule information is included.	2. Most phonological rules are not easily applied to diphones.
Syllable	1. It is relatively easy to locate and identify.	1. Precise syllable boundaries are difficult to determine.
	2. It includes much coarticulation rule information.	2. The total number of syllables can be rather large.
	3. Certain phonological rules include syllable boundary conditions.	
Word	1. It eliminates an entire level of recognition activity.	1. Template matching is more difficult with large vocabularies.
		2. Junctural phonological rules are hard to characterize in lexicon entries.

Tableau II.1 : Résumé des avantages et des désavantages de chaque unité phonologique qui peut être utilisée en reconnaissance reconnaissance automatique de la parole.
Tiré de (Lea 80).

2.4.1.2 Analyse lexicale.(Lea 80).

Présentons aussi rapidement la fonction de l'analyse lexicale évoquons les méthodes utilisées et les problèmes pour retrouver les mots présents dans l'énoncé.

Deux fonctions sont habituellement distinguées, à savoir: l'émission d'hypothèses sur les mots et la vérification de mots. Le terme "mot" est ici employé pour désigner des mots de la langue mais aussi des syntagmes considérés comme irréductibles tels que " au fur et à mesure ", " c'est à dire " ... .

L'émission d'hypothèses sur les mots:

Cette opération correspond à l'établissement d'une liste de mots pouvant éventuellement être présents à un endroit du signal, en comparant une description théorique des mots qui se trouve dans un lexique et la forme du signal à cet endroit.

Il est nécessaire de souligner qu'il n'y a jamais de "mapping" complet entre l'information acoustique et un mot de vocabulaire admis. Le bruit parasite venant de l'environnement, les différences entre les locuteurs, les différences pour un même locuteur à différents moments, les variations de prononciation rendent difficiles la reconnaissance des mots présents dans l'énoncé. De plus, lorsqu'il s'agit d'un discours continu, la représentation acoustique d'un mot est intégrée dans la phrase et modifiée par elle en raison de problèmes de coarticulation.

En général, un auditeur reconnaît les différents mots d'un texte non seulement par ce qu'il entend mais également par le contexte et grâce à ce qu'il s'attend à entendre.

Vérification de mots:

La deuxième opération consiste à examiner une liste de mots supposés être présents à un endroit particulier de l'énoncé, et à déterminer lesquels ressemblent le plus au signal à cet endroit. On leur attribue un score suivant leur ressemblance par rapport à cet endroit du signal.

Cette liste d'hypothèses de mots peut provenir soit des niveaux supérieurs (syntaxiques et sémantiques) qui à l'aide des contraintes sur les structures syntaxiques et sémantiques peuvent supposer qu'un mot est présent, soit des hypothéses sur les mots. Ce niveau fait donc le lien entre les connaissances que l'on a de la langue et de l'application (syntaxe, sémantique, pragmatique) et les connaissances indépendantes de l'application (acoustique,

phonétique, phonologiques).

On aura donc 2 sous-modules correspondant à ces fonctions :

L' "hypothétiseur" de mots.

Son effet est de sélectionner parmi l'ensemble des mots liés à une application, ceux qui ont une grande probabilité d'être présents à un endroit de l'énoncé. En fait, son but premier est de rendre la reconnaissance des mots présents dans l'énoncé plus rapide. Les performances de cet hypothétiseur (vitesse de travail, nombre d'hypothèses fausses, d'hypothèses exactes …) sont déterminantes pour la réduction du temps de reconnaissance de l'énoncé.

Bien que différentes méthodes existent pour déterminer la liste des mots-hypothèses, toutes réalisent une même opération qui est de prendre la description acoustique d'un mot de vocabulaire, de prendre la partie du signal considérée et de sortir tous les mots présentant une similarité suffisante avec la partie du signal choisie. Ces méthodes diffèrent par la manière de parcourir l'espace des mots du vocabulaire.

Il existe parmi d'autres:

Le matching itératif : l'espace des mots est d'abord parcouru rapidement et avec peu de précautions pour repérer les mots qui ressemblent grossièrement avec la partie du signal choisie. Ensuite, cette sous-région est reparcourue avec plus de soins pour déterminer les mots ressemblant davantage avec la partie du signal choisie. Ensuite, itérativement et avec de plus en plus de soins, le processus recommence jusqu'à trouver le ou les mots les plus corrects possibles.

Le data-driver matching : cette méthode utilise des informations acoustiques pour contrôler pas à pas la recherche à travers une structure de données combinant les descriptions de tous les mots de vocabulaire. Durant la recherche, ce sont des sous-parties de mots qui sont comparées avec la partie du signal

choisie et non tous les mots comme dans la méthode précédente. L'information acoustique peut alors être vue comme un index de la structure choisie. Comme index, l'information acoustique la plus utilisée se trouve sous forme de phonèmes ou de labels de segments. Cette deuxième méthode est la plus souvent utilisée.

Le vérificateur de mots.

Son objectif est de déterminer si les mots présents dans une liste de mots le sont également à un endroit donné du signal. Il donnera à chacun de ces mots un score. Cette évaluation se fera en fonction des informations acoustiques et phonétiques dont le vérificateur dispose. Ces informations proviennent d'un lexique contenant la définition théorique des mots et de leur règles contextuelles qui reflètent les modifications que ces mots peuvent subir lorsqu'ils sont situés à l'intérieur d'une phrase.

2.4.1.3 L'analyse syntaxique et l'analyse sémantique. (Lea 80, Pierrel 81).

Dans le but de comprendre une phrase, on a besoin d'une analyse de sa syntaxe mais surtout d'une analyse de sa sémantique. En effet, l'analyse sémantique s'occupe de la signification de la phrase et c'est ce qui est important si on désire comprendre cette phrase.

Le but de l'analyse syntaxique est de déterminer la syntaxe de la phrase (en même temps de vérifier si elle est correcte syntaxiquement) et de créer une représentation syntaxique de celle-ci. Cette structure étant appelée la *structure de surface* de la phrase.

Le but de l'analyse sémantique est de réaliser l'interprétation de l'énoncé du point de vue de son sens et cela de façon non ambigüe et adéquate. La représentation sémantique qui sera construite sera appelée *structure profonde* de la phrase.

Ces deux structures syntaxique et sémantique serviront alors d'input au niveau suivant qui les interprètera dans le contexte

d'un dialogue avec le locuteur.

Il est bien évident que de nombreux modèles de définition et de représentation des structures de phrase peuvent être adoptés. Certains possèdent un caractère purement syntaxique, d'autres font usages de données complémentaires de type sémantique ou pragmatique. On trouvera alors principalement des modèles purement syntaxiques, des modèles syntaxico-sémantiques et des modèles lexicaux. Ces modèles de grammaire seront développés dans le chapitre suivant.

2.4.1.4 Analyse sémantico-pragmatique { Lea 80, Haton 84, Pierrel 81 }

Cette dernière partie d'analyse consiste principalement à préciser l'interprétation du point de vue du sens de l'énoncé dans le cadre du contexte fixé. Cela parce qu'une phrase peut avoir un sens différent suivant le contexte dans lequel elle est énoncée. On a besoin pour réaliser cette opération de diverses sources de connaissances. Il y a les connaissances sémantiques qui permettent de connaître le sens global de la phrase et les connaissances pragmatiques qui définissent le contexte de l'application et qui permettent de comprendre exactement le sens de la phrase par rapport au dialogue antérieur et à l'application. Ce contexte courant se compose d'une partie fixe qui sont les connaissances sur l'application et des connaissances variables modifiées progressivement par les renseignements apportés par le dialogue.

Ce domaine d'analyse en étant encore au début de son développement, les solutions apportées dépendent fortement de l'application et des réalisateurs des systèmes. Néanmoins, on peut tirer quelques enseignements généraux sur les fonctions de cette analyse.

Elle réalise, d'après { Lorant 86 }:

" - l'établissement de la représentation sémantique de l'énoncé, ainsi que son interprétation.
Il est nécessaire que ce module possède un certain nombre de

connaissances. Une partie de celles-ci proviendra du lexique où se trouve une définition sémantique des mots. Il aura besoin également d'un formalisme de représentation de la sémantique et des relations qui peuvent exister entre les divers concepts ainsi que d'un modèle du monde dans lequel l'application se déroule.

- Le rejet des énoncés qui sont syntaxiquement valables mais qui sont sémantiquement inacceptables dans le cadre de l'application choisie. "

2.4.1.5 La prosodie. (Lea 80)

Il reste un domaine d'analyse qui est fort peu utilisé et qui semble pourtant fort prometteur pour l'aide à la reconnaissance des phrases par la machine. C'est celui de la prosodie.

La prosodie a pour but d'utiliser les informations telles que les intonations, les pauses, les structures temporelles de l'énoncé de la phrase. L'analyse prosodique pourrait améliorer certainement les performances des systèmes de compréhension de la parole.

La prosodie est importante car elle est totalement indépendante des autres niveaux d'analyse et ne se trouve donc pas influencée par les erreurs contenues dans les résultats de ceux-ci. Elle peut de plus aider fortement ces autres analyses par ses résultats. Un exemple peut nous éclairer de la manière dont l'information prosodique peut être utile dans les systèmes de compréhension de la parole.

Elle peut aider à séparer les différents mots d'une phrase grâce aux respirations et à l'intonation du locuteur. Soit dans l' exemple classique qui est celui des vers de Victor Hugo :

" Gal, amant de la reine, alla, tour magnanime
galament de l'arène à la tour Magne à Nîmes. "

ou d'autres plus usuels tels que :

L'apesanteur et la pesanteur; la mousse tache et la moustache;

le maladroit, le mâle a droit et le mâle adroit; le voleur dévalise et le voleur des valises, ...

La prosodie peut aussi :

- aider à détecter le type de phrase (affirmative, interrogative, exclamative , ...). Cela peut se faire en étudiant l'intonation c'est-à-dire la variation de l'amplitude moyenne du signal tout au long de la phrase.

- détecter acoustiquement certains aspects de la structure syntaxique. Et cela sans dépendre des séquences de mots hypothétiques (et qui contiennent potentiellement un taux d'erreurs certain), dérivés des informations acoustiques et phonétiques. Par exemple, pour différencier les 2 énoncés suivants : " What is on the road ahead ? "
  " What is on the road ? A head ? "

- aider à choisir quelle règle appliquer préférentiellement dans la réalisation des différentes analyses. Par exemple, beaucoup de règles phonologiques demandent des informations pour savoir si une syllabe est étirée ou contractée. Cette information peut être détectée en analysant la forme du signal.

- aider à ajuster l'ordre de priorité des listes des mots hypothétiques. Certains mots sont habituellement prononcés avec une certaine insistance, ou sont précédés par des vallées (fortes diminutions suivies de fortes augmentations). Par exemple, les verbes principaux, les noms, les adjectifs sont souvent renforcés tandis que d'autres mots tels que les prépositions ou les articles ne le sont pas ou peu. Si on détecte une insistance sur un mot et que la liste des mots hypothèses contient un verbe, le score de ce verbe, correspondant à sa probabilité de présence, va augmenter.

- vérifier ou modifier les scores des structures déjà déterminées à partir des informations phonétiques.

- aider à séparer les différentes propositions de la phrase.

etc ...

Malheureusement, malgré tous les avantages qu'elle pourrait apporter, la prosodie est encore fort peu utilisée. Elle ne l'est souvent que pour vérifier ou modifier les hypothèses venant des autres niveaux d'analyse.

2.4.2 Les stratégies de contrôle et les interactions des diverses sources de connaissances. (Lea 80, Lorant 86).

Après avoir décrit les différents types d'analyse habituellement utilisées dans les systèmes de compréhension de la parole, on peut alors se demander dans quel ordre il va falloir les activer et puisque les résultats de l'une doivent servir à l'autre, il faudrait également savoir comment ils vont transiter d'un module d'analyse à un autre.

2.4.2.1 Interactions des sources de connaissances.

Pour le type d'interaction de ces différentes analyses, plusieurs solutions peuvent être adoptées. Citons celles qui sont le plus couramment utilisées.

- le modèle hiérarchique (ou ascendant).
- le modèle goal-oriented (ou descendant).
- le modèle hétérarchique.
- le modèle "blackboard".

1°) Le modèle hiérarchique.

Dans ce modèle, les données voyagent de niveau à niveau, à partir des niveaux inférieurs vers les niveaux supérieurs. Les résultats d'un module sont disponibles uniquement pour le module qui lui est immédiatement supérieur. Il n'y a pas de communication

d'un niveau supérieur vers un niveau inférieur. Cela entraîne qu'une erreur dans le résultat d'un niveau inférieur est quasiment impossible à corriger au niveau supérieur car il ne peut pas demander de renseignements complémentaires au niveau inférieur.
Voir figure II.3.a.

2°) Le modèle goal-oriented.

Ici, la communication commence aux niveaux les plus hauts. Ces niveaux établissent des prédictions pour les niveaux inférieurs jusqu'à ce que le niveau de base soit atteint. A ce moment, la prédiction est confirmée ou infirmée ou encore, plus généralement, un score lui est donné. Les évaluations des différentes prédictions se font alors en sens inverse de module en module vers le niveau supérieur. Ce modèle possède l'inconvénient que l'espace de recherche initial du module supérieur est relativement large.
Voir figure II.3.b.

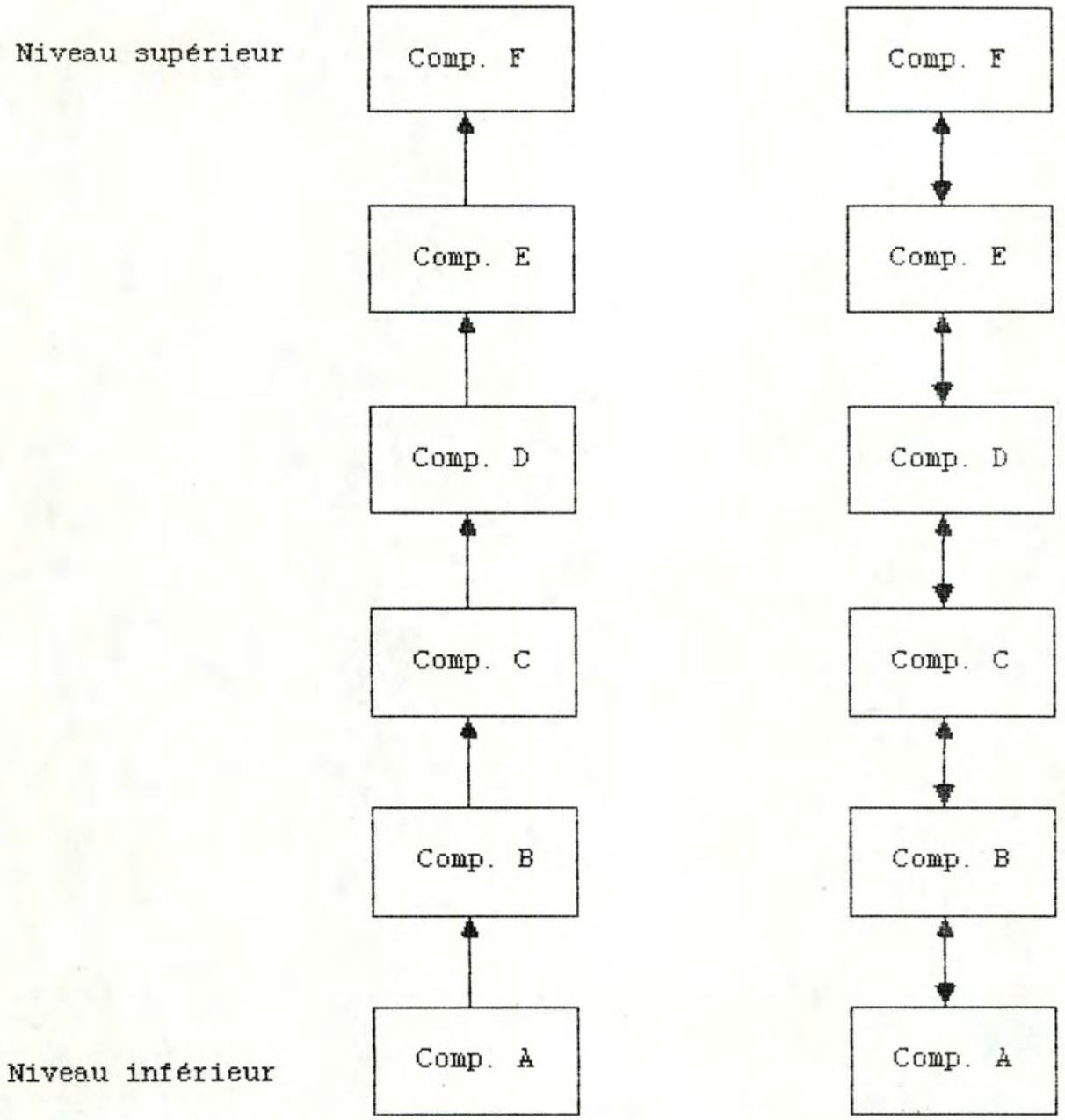
3°) Le modèle hétérarchique.

Dans ce cas, toutes les sources de connaissances peuvent interagir entre elles. Le prix à payer est celui d'une plus grande complexité, en particulier chaque canal de transfert de données entre deux modules exige une représentation commune des informations. Cela augmente la complexité de chaque module.
Voir figure II.3.c.

4°) Le modèle Blackboard.

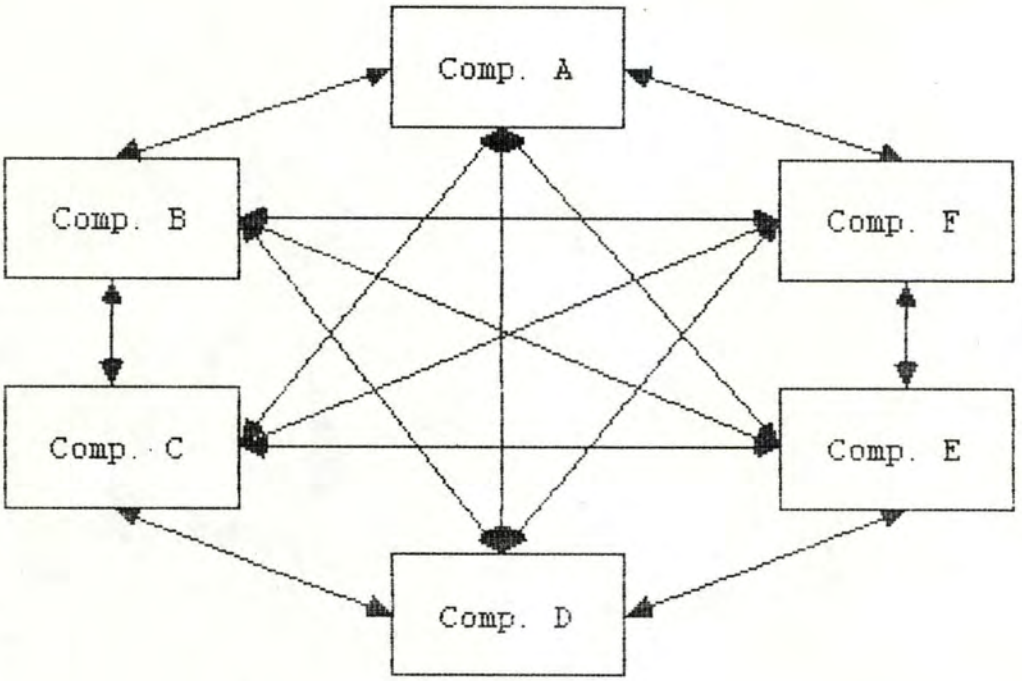
Toutes les sources de connaissances interagissent à travers une base de données centrale. Chaque source est indépendante. Elle examine la banque de données, peut alors évaluer les différentes hypothèses créées par les autres et y apporter les siennes. Malheureusement, le contrôle de la recherche avec ce modèle reste difficile { Lea 80 }.
Voir figure II.3.d.

Cette liste est loin d'être exhaustive et généralement, on utilise des combinaisons de ces modèles car tous possèdent des

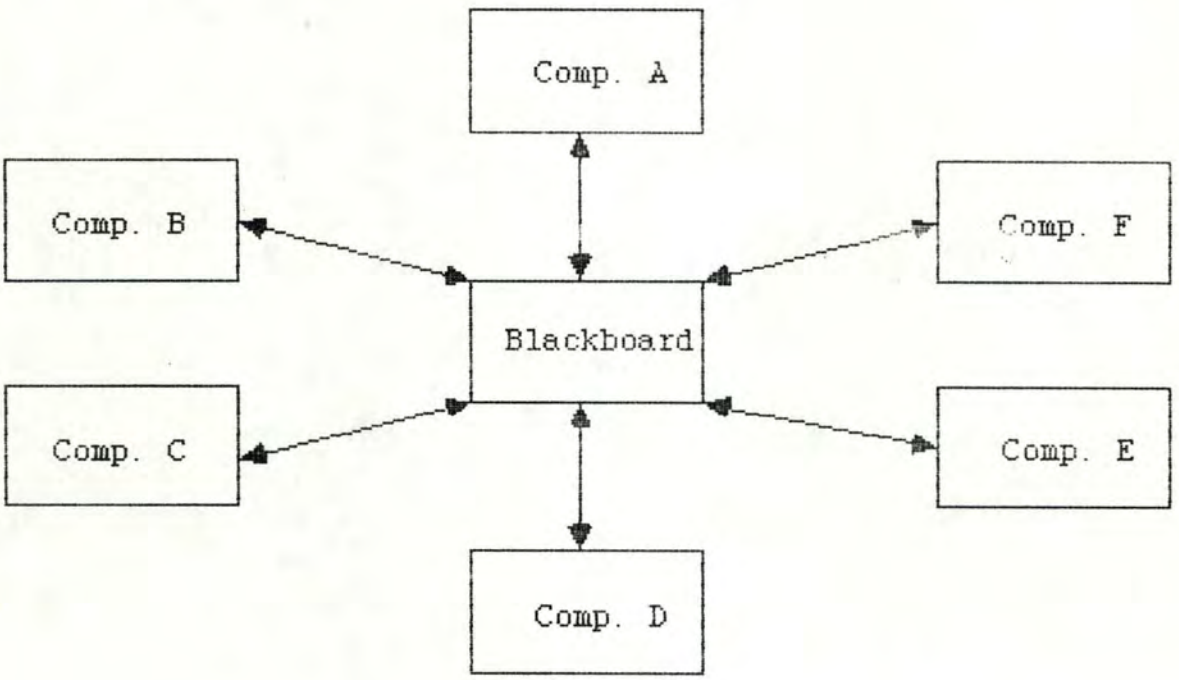


a) Modèle hiérarchique b) Modèle goal-directed

Figure II.3 : Interaction des sources de connaissance.



c) Modèle hétérarchique.



d) Modèle Blackboard.

Figure II.3 : Interaction des sources de connaissance.

avantages et des inconvénients.

Il reste alors à savoir dans quel ordre ces différentes analyses vont se réaliser lors de la reconnaissance d'un énoncé..

2.4.2.2 Les stratégies de contrôle.

Comme à l'habitude, il y a plusieurs possibilités. On peut , par exemple, développer complètement une première hypothèse c'est-à-dire lui appliquer tous les types d'analyse, avant d'envisager d'en développer une autre. Cette stratégie est appelée en profondeur d'abord. Une autre solution serait de développer toutes les solutions possibles pour une analyse avant de les soumettre à l'analyseur suivant. Cette stratégie est appelée en largeur d'abord. Ou encore, on peut imaginer une approche intermédiaire entre ces deux solutions : on penserait alors à ne développer que l'une ou l'autre solution, celles dont le niveau de plausibilité dépasse un certain seuil. Cette stratégie s'appelle les quelques meilleurs d'abord.

Signalons encore que lors de l'analyse d'un énoncé, il faut savoir par ou commencer. Différentes méthodes sont également possibles. Soit analyser l'énoncé de la gauche vers la droite, soit de la droite vers la gauche, soit à partir d'ilots de confiance détectés dans l'énoncé (segments de l'énoncé qui sont reconnus avec une grande probabilité) et de les développer ensuite latéralement.

Celui qui souhaiterait approfondir ces recherches pourra éventuellement se reporter à la bibliographie.

2.5 Conclusion.

Ainsi donc, il existe diverses manières d'analyser un énoncé : sans doute sont-elles aussi nombreuses que les différentes analyses possibles. Les analyses présentées sont celles que l'on utilise le plus couramment utilisées. Pour réaliser ces analyses, il n'y a

pas non plus de méthode unique. Chaque système choisit les méthodes qu'il estime être les plus efficaces. Mais en plus, chaque nouveau système apporte une collaboration originale et des connaissances nouvelles au problème de la reconnaissance et de la compréhension de la parole. Il peut, par exemple, présenter une nouvelle manière de modéliser certaines connaissances (la grammaire, le lexique, …), une nouvelle méthode plus rapide, plus efficace pour réaliser une analyse, … Ce qui vient d'être expliqué dans ce chapître ne donne qu'une idée générale des analyses et des méthodes les plus fréquemment utilisées. Il permet de comprendre les mécanismes généraux qui sont les plus employés dans ce domaine de recherche.

Nous pouvons dans le chapître suivant, nous attarder sur certaines techniques de l'analyse syntaxique et de l'analyse sémantique, car c'est dans cette partie que se situe mon apport au système en cours de développement au CRIN à Nancy.

Chapitre 3.

Les grammaires utilisées pour construire les représentations syntaxiques et sémantiques des énoncés.

Examinons, plus en détails, le niveau de l'analyse syntaxico-sémantique. Nous parlerons des grammaires qui sont employées pour analyser la structure des langages dans diverses applications,puis présenterons différents modèles de grammaires et nous attarderons à celles qui sont implémentées dans le système de dialogue en construction à Nancy.

La grammaire est, d'après (Grévisse 69), l'étude systématique des éléments constitutifs d'une langue. Elle comprend :

1° : La phonétique ou science des sons d'une langue.

2° : La lexicologie ou science des mots. A la lexicologie se rapporte entre autres la *sémantique*, science de la signification des mots.

3° : La syntaxe ou ensemble des règles qui régissent l'arrangement des mots et la construction des propositions.

Ayant précédemment donné un aperçu de l'analyse phonétique et lexicale. J'en arrive maintenant à discuter plus en détails l'analyse syntaxique et l'analyse sémantique par la machine.

Les grammaires définissent les structures de phrases (les composants, les relations entre ces composants des phrases) qui sont admises dans le langage. Il est donc important de connaitre ces grammaires.Dans ce cadre, on peut discerner 2 problèmes étroitement mêlés : celui du formalisme adéquat pour modéliser la grammaire et ensuite, celui de la détermination d'un ensemble de règles permettant d'utiliser cette grammaireselon l'une des 2 manières suivantes : soit pour construire une phrase (-> aspect génératif), soit pour vérifier si une phrase est correcte par rapport à cette grammaire et construire éventuellement une représentation de la structure de la phrase (-> aspect d'analyse).

Remarquons que lorsque l'on parle de la structure syntaxique

de la phrase, on parle de sa *structure de surface*, tandis que lorsque l'on parle de sa structure sémantique, on parle alors de sa *structure profonde*.

Dans le cas de la compréhension de la parole, c'est surtout cette représentation sémantique qui nous intéresse. En effet, celle-ci tient compte de la signification des mots et c'est ce qui est important dans si on veut comprendre ce que le locuteur demande. Mais l'analyse syntaxique n'est pas pour autant à négliger. Le but de cette analyse est de fournir des informations structurelles, au sujet de l'énoncé, au composant d'analyse sémantique qui lui, tentera alors de déterminer la structure profonde de la phrase. L'analyse syntaxique est intéressante parce qu'elle limite le nombre de suppositions que le module d'analyse sémantique doit faire pour réaliser son analyse. Le nombre de représentations sémantiques possibles de la phrase est grâce à cela plus réduit.

Pour l'analyse de la syntaxe et de la sémantique, différents modèles de grammaire ont été développés.
Pierrel { Pierrel 81 } dégage de la littérature différents types de modèles que l'on peut classer en :

- Les modèles purement syntaxiques.
- Les modèles syntaxico-sémantiques.
- Les modèles lexicaux et/ou sémantiques.

Approfondissons ces modèles et regardons de plus près les grammaires les plus couramment utilisées. Il n'est pas dans mon intention de présenter une liste complète des grammaires existantes. Ce serait un travail bien fastidieux et d'ailleurs inutile. Il suffit de se référer à la littérature déjà existante si on désire des informations complémentaires. Nous nous appuyons principalement sur { Pierrel 81, Deville 86, Bruce 75, Lorant 86, Fill 68, ... }.

3.1 Les modèles purement syntaxiques.

Ces modèles ne pourront pas être employés seuls dans notre système puisque nous sommes intéressés par la structure profonde de la phrase. Il devront donc être complétés si on veut les utiliser. Néanmoins, il n'est pas inutile d'y jeter un coup d'oeil pour nous donner une première idée des représentations grammaticales des phrases et des méthodes d'analyse.

Les plus connues de ces grammaires sont les *grammaires hors-contextes de Chomsky* (Chom 71).

Elles sont composées

- d'un ensemble de symboles de base pouvant être
  - . des terminaux de la grammaire (nom propre, préposition , pronoms, ...)
  - . des non-terminaux tels que
    - < GN > (groupe nominal)
    - < PRR > (proposition relative)
    - < PREP > (préposition)

- d'un ensemble de règles du type
  - X ::= Y
  - se lisant X est composé de Y
  - où X est un non-terminal de la grammaire
  - Y est une suite de terminaux et/ou de non-terminaux.

Par exemple, la figure III.1. donne la définition d'un groupe nominal tel qu'on souhaiterait l'obtenir dans le système MYRTILLE 2, on a la .

Pour ces modèles de Chomsky, une présentation visuelle, sous forme de réseau permet de saisir plus facilement le processus. la figure III.2. donne l'analyse d'un groupe nominal dont la figure III.1 énonçait les règles.

< GN > ::= nom propre
::= < SN >
::= < SN > < PRR > < RELAT >
::= < SN > < PREP1 > < GN >

< PRR > ::= qui | que | ...

< SN > ::= < DET > < NOMQ > | < NOMQ >

< NOMQ > ::= < NOMQD > | Adjectif < NOMQD >
::= < CHIF > < NOMQD >

< NOMQD > ::= nom | nom adjectif

< DET > ::= < ART > | < ADJINT >

< ART > ::= le | la | les | des | un | ...

< ADJINT > ::= quel | combien de | ...

< PREP1 > ::= de | sur | ...

< RELAT > ::= .....

Figure III.1 : Exemple de règles du modèle de grammaire hors-contexte de Chomsky. Ici, exprimées dans le formalisme de Bakus-Naur. Tiré de (Pierrel 81).

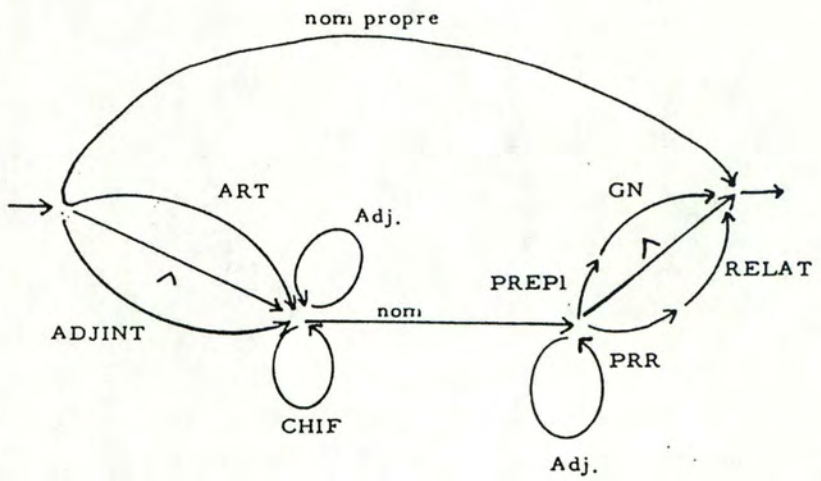
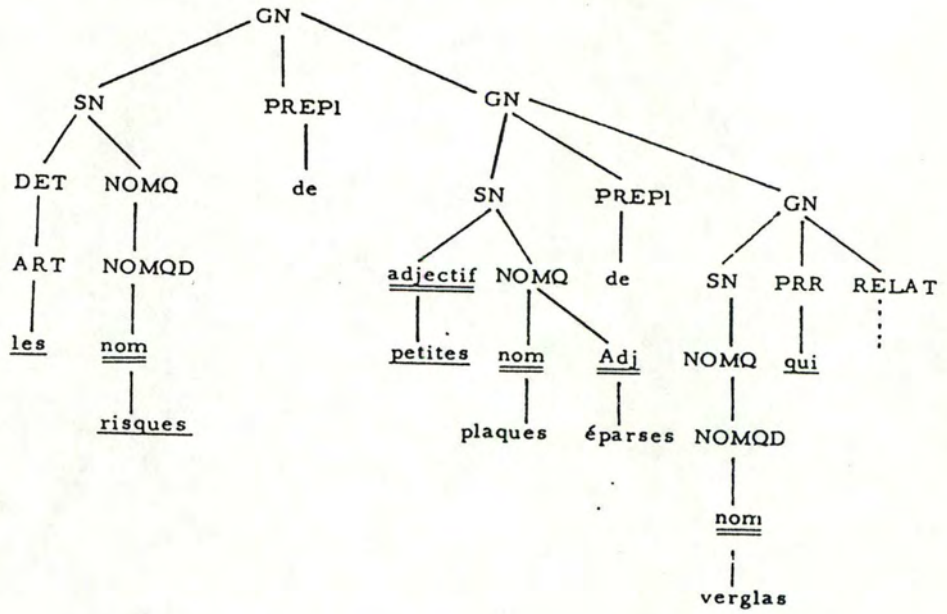


Figure III.2 : Réseau représentant la grammaire du groupe nominal. Tiré de (Pierrel 81).



GN : représente un nom terminal de la grammaire

nom : un terminal de type sous lexique

les : un terminal vrai

Figure III.3 : Exemple d'arbre syntaxique. Tiré de (Pierrel 81).

Ce style de réseau donne une représentation syntaxique de l'énoncé sous forme d'arbre créé au fur et à mesure du choix des règles. La figure III.3. permet de représenter, par exemple, le groupe nominal " les petites plaques éparses de verglas ".

Ces réseaux portent le nom de R.T.N. (Recursive Transition Network). On remarque que la définition de la grammaire et des règles sont ici mêlées.

Une grammaire de ce type peut s'utiliser sous son aspect analytique ou sous son aspect géneratif .

Les avantages de ces grammaires ne sont pas négligeables. Pierrel cite entre autres dans { Pierrel 81 } :

" - des procédures d'analyse bien spécifiées.
- la possibilité d'obtenir automatiquement la structure de la phrase analysée.
- une définition de la structure indépendante du vocabulaire utilisé.

Ses principaux désavantages sont :

- la lourdeur d'une telle grammaire si on veut rendre compte des nombreuses possibilités du langage parlé.
- l'inadaptation d'un modèle génératif pour effectuer une reconnaissance.
- la trop grande importance prise par les petits mots. "

Dans ce type de modèle, on peut également mentionner les *grammaires en chaînes de Harris* et aussi des modèles plus compliqués tels les grammaires transformationnelles de Chomsky que nous allons évoquer.

Les grammaires transformationnelles de Chomsky.

Ces grammaires possèdent l'avantage de, à partir de phrases de même structure profonde mais de structure de surface différente,

pouvoir reconstruire un même arbre syntactique pour ces différentes phrases.

Il permet également de passer d'une représentation sémantique d'une phrase vers une représentation syntaxique de celle-ci. Malheureusement, ce type de grammaire est de type génératif et ne permet guère que de construire les phrases. Il convient bien pour la génération automatique de phrases mais non pour l'analyse automatique (Woods 75). Notre intérêt est en fait de découvrir la structure profonde d'une phrase et cela à partir de sa représentation syntaxique : c'est-à-dire que ce type de grammaire nous offre l'inverse de ce que nous cherchons.

3.2 Les modèles syntaxico-sémantiques.

Le principe de base des modèles syntaxico-sémantiques est de faire intervenir la sémantique en lien avec la syntaxe pour définir le langage naturel.

3.2.1 Les grammaires sémantiques.

Elles sont calquées sur les grammaires hors-contextes de Chomsky mais où les classes syntaxiques telles que nom, verbes, prépositions, ... sont remplacées par des classes sémantiques telles que mesures, verbes de tel type, ... Le problème principal de ces grammaires est de lier très fortement la définition de la grammaire et les règles de l'analyseur. Chaque application possédant sa grammaire propre, il faut pour chacune de celles-ci redéfinir également les règles de l'analyseur. Ce qui constitue un manque de généralité.

3.2.2 Les grammaires systémiques.

Ces modèles font apparaître dans une sorte de graphe les différents cas possibles (les systèmes) et les décisions à prendre lors de l'analyse d'une phrase.

On peut mentionner parmi les réalisations :

1°) les *réseaux de transition de Woods* (Woods 70), appelés les A.T.N. (Augmented Transition Network).

Ces réseaux se composent de noeuds et d'arcs qui représentent des états et des transitions. Ils ressemblent aux R.T.N. auxquels on aurait ajouté (Deville 86) :

" - des tests sur les arcs pour savoir s'il convient de les emprunter.

- des actions de construction de la structure associés aux arcs.

- des registres aux arcs, améliorant les performances de l'analyseur. "

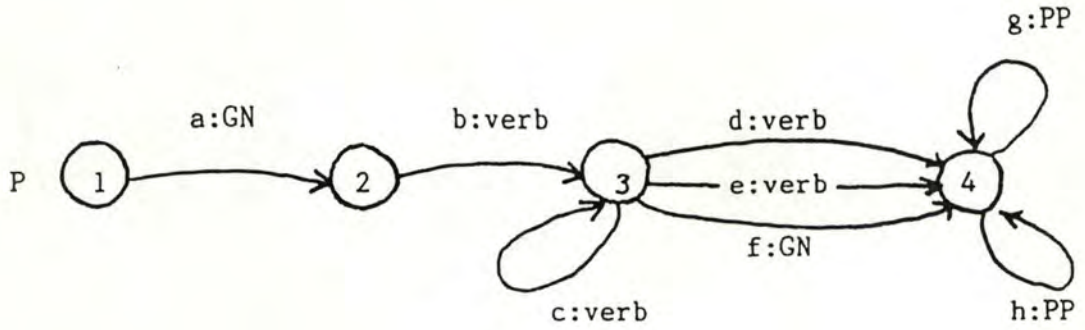
Un exemple d'A.T.N. tiré de (Arte 84) est présenté en figure III.4.

Une autre réalisation de ce type de grammaire est :

2°) Les *réseaux à noeuds procéduraux (R.N.P) de Pierrel.* (Pierrel 81)

Une nouvelle évolution a été réalisée par J-M. Pierrel. Le principal avantage de ces R.N.P. par rapport aux A.T.N. de Woods réside dans la séparation nette entre le modèle de la grammaire et les règles des procédures d'analyse de l'itinéraire. Cette séparation entre l'analyseur et la grammaire induit un autre avantage : chaque fois que l'on désire changer la grammaire pour pouvoir représenter un autre sous-langage, il n'est pas nécessaire de modifier l'analyseur, ce qui est le cas dans les A.T.N. où l'analyseur est fortement lié à la grammaire.

Pierrel a développé un système similaire aux A.T.N. mais plutôt que de placer les règles d'analyse de parcours sur les arcs, il les a placés sur les noeuds. Ces noeuds ont alors un rôle triple :



a:GN
    action : put the current word in the "subject" register

b:verb
    action : put the current word in the "main verb" register

c:verb
    condition : if the main verb is 'être' or 'savoir'
    action : put the contents of "main verb" in the "auxiliary" register
    and put the current word in the "main verb" register

d:verb
    condition : if the current word is a past participle,
    and the "main verb" is 'être'
    and the current word is a transitive verb
    action : put the label "passive" in the "mood" register,
    transfer the contents of "main verb" to "auxiliary",
    put the current word in the "main verb" register

e:verb
    condition : if the current word is an "infinitive",
    and the "main verb" is a "mental process" (vouloir, dire,...)
    action : put the current word in the "direct object" register

f:GN
    action : put the resulting structure from the GN analysis in the
    "direct object" register

g:PP
    condition : if the "mood" is "passive"
    and the "preposition" register of the structure analysed
    by PP is the word 'par'
    action : place the value of the "object preposition" register of this
    structure in the "subject" register

h:PP
    action : put the analysed structure of PP in the
    "complement" register.

Figure III.4 : Exemple de A.T.N.
Tiré de (Arte 84).

- prendre en compte les phénomènes de type contextuels.
- traiter les mots courts de liaison.
- réduire en grande partie l'indéterminisme du réseau en triant les sorties possibles d'une procédure, compte tenu des entrées, du contexte déjà traité et des résultats de la procédure en cours.

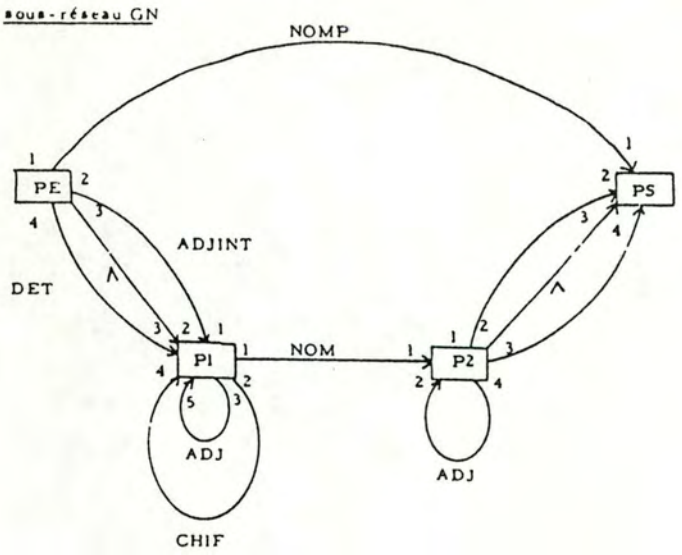
Une représentation en graphe de ces R.N.P., similaire à celle des A.T.N., est donnée dans le cas du groupe nominal dans la figure III.5.

Ces R.N.P. sont un modèle linguistique pour un analyseur syntaxique d'un sous-langage. Les tests réalisés sur les R.N.P. lors de leur implémentation dans le système MYRTILLE 2 ont permi de montrer leur validité et leur robustesse comme modèle de définition d'un langage pseudo-naturel. C'est pourquoi c'est le modèle que nous allons utiliser pour créer la structure syntaxique de la phrase dans le système qui est actuellement en cours de développement au CRIN.

Pour plus d'informations sur les R.N.P., l'annexe fournit la définition des procédures utilisées dans le cadre d'une application de renseignements météorologiques implantée dans le système MYRTILLE 2. Des informations détaillées sont présentes dans { Pierrel 81 }.

3.2.3 Les " grammaires de cas ".

Ce modèle de grammaire se base sur l'observation de Fillmore qu'il n'est pas nécessaire de réaliser une analyse complète de la phrase dans un système de compréhension de la parole. On peut ne s'intéresser qu'aux composants principaux de la phrase ainsi qu'à un ensemble d'attributs syntaxico-sémantiques portant sur les relations (les cas) entre ces composants. Ces attributs sont par exemple , le temps (présent, passé, futur), la voix (active, passive), la forme (affirmative, négative). Les relations peuvent être du type " agent d'une action ", " objet d'une action", " bénéficiaire d'une action ", ...



PE : Procédure d'entrée

PS : Procédure de sortie

P1 : } Procédures internes
P2 : }

ADJINT : Adjectif interrogatif (accès au lexique)

ADJ : Adjectif (accès au lexique)

CHIF : Chiffre (référence à un sous réseau)

DET : Déterminant (accès au lexique)

GN : Groupe nominal (référence à un sous-réseau)

NOM : nom (accès au lexique)

NOMP : nom-propre (accès au lexique)

RELA : proposition relative (référence à un sous réseau)

Λ : branche vide

```
Procedure P2 of GN :

Begin (* P2 *)
  stack (exit 2)
  if not end lexical lattice then
    begin
      if entrance = 1 then stack (exit 4)
      if POSSIB (complement with "sur") and PRESENCE (sibilant)
         then stack (exit 3)
         else
           begin
             if PRESENCE (plosive) then
               if sonorant then
                 begin
                   stack (exit 2)
                   stack (exit 3)
                 end
               else
                 begin
                   stack (exit 3)
                   stack (exit 2)
                 end
           end
    end
end (* P2 *)
```

(Pierrel 1981:166)

Figure III.5 : a) R.N.P. du groupe nominal dans Myrtille II.
b) Exemple de procédure R.N.P.
Tiré de (Pierrel 81).

Une grammaire de cas peut être employée pour étudier la structure de surface d'une phrase mais aussi sa structure profonde .

Examinons 2 grammaires de cas : celle de Fillmore et celle de G. Deville et H. Paulussen. De la première, je ne donnerai qu'un rapide aperçu et je m'attarderai en particulier sur la seconde car c'est cette grammaire qui sera utilisée dans notre système pour représenter la structure profonde d'un énoncé.

Intuitivement, on peut comparer la notion de cas à celle utilisée dans différents langages tels que le latin, le grec ou l'allemand où les différents noms et propositions s'accordent en fonction de leur cas, c'est-à-dire de leur fonction dans la phrase (sujet, objet, ... ou nominatif, accusatif, ...).

1°) La grammaire de cas de Fillmore. (Fill 68).

Fillmore définit la notion de cas comme une relation sémantique entre un prédicat (souvent un verbe) et ses arguments. On peut considérer que la théorie de Fillmore fournit pour chaque phrase une structure consistant en un verbe, ses modalités et ses arguments qui sont les différents cas.

Un exemple de représentation d'un énoncé développé à partir de la grammaire de cas de Fillmore est donné en figure III.6.

Un reproche a souvent été fait à l'encontre de cette grammaire : les cas ne sont pas toujours définis par rapport au prédicat, ce sont parfois uniquement des simples concepts sémantiques. Par exemple, le cas objectif de la figure III.6 qui sert à tous les constituants pour lesquels une fonction sémantique précise par rapport au prédicat n'a pu être assignée.

D'autres grammaires de cas ont été développées par *Schank* (Schank 72) , *Dik* (Dik 79) et bien d'autres. Toutes ces grammaires diffèrent par la définition qu'elles donnent du prédicat, des cas et des relations entre les prédicats et les cas.

Je ne peut pas reprendre en détails toutes ces grammaires. Bruce { Bruce 75 } réalise un large aperçu de ces diverses grammaires de cas développées jusqu'en 1975. Regardons uniquement de plus près celle que nous implémenterons dans notre système, à savoir la grammaire de cas développée par G. Deville et H. Paulussen. J'en reprends ici les caractéristiques principales tirées de { Deville 86 }.

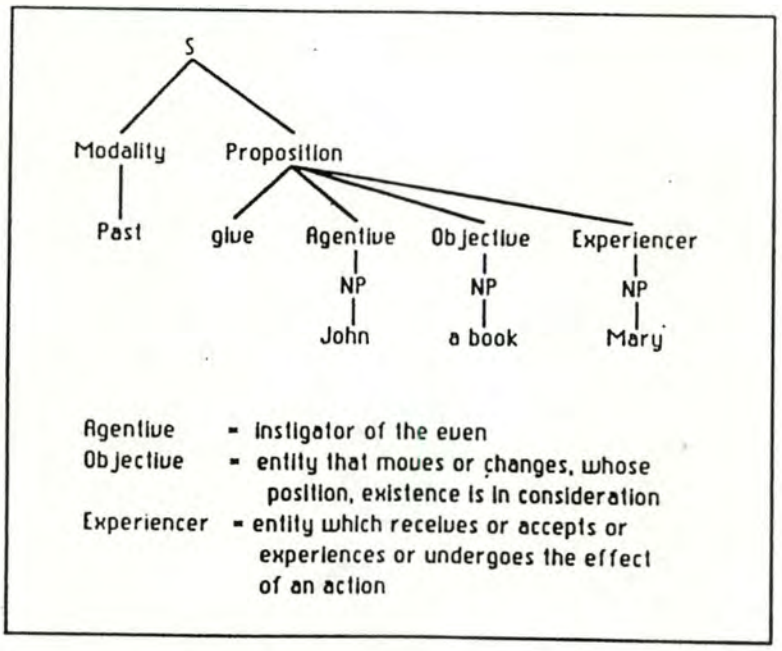


Figure III.6 : Représentation de la structure profonde dans la grammaire de cas de Fillmore.

2°) La grammaire de cas de G. Deville et H. Paulussen.

G. Deville et H. Paulussen proposent une grammaire fournissant une représentation de la structure profonde d'une phrase. Cette représentation exprime les relations sémantiques qui existent dans la phrase entre un prédicat et ses arguments.

Un prédicat est l'élément pivot d'une expression linguistique. Un prédicat peut être un verbe, un nom ou un adjectif. Par exemple, " donner ", " don ", " valide ".

Tous ces prédicats sont dérivés d'un ensemble fini de primitives et tous les prédicats d'une même primitive possèdent un certain nombre de propriétés communes. Par exemple :

Primitive	Exemple	Propriétés
Mouvement 1	aller	. se réfère à au moins 1 entité animée. . verbe de mouvement. . verbe intransitif.
Mouvement 2	apporter	. se réfère à au moins 1 entité animée. . verbe de mouvement. . verbe transitif.
Echange-production	donner	. se réfère à au moins 1 entité animée. . pas de mouvement. . verbe transitif. . une entité est transférée d'une entité à une autre.
Location 2	garder	. se réfère à au moins 1 entité animée. . pas de mouvement. . verbe transitif. . doit faire référence à une dimension d'espace.
Process 1	échanger	. pas de mouvement. . verbe transitif.

se réfère à au moins 1 entité allumée.

Pour une liste exhaustive des primitives, consulter { Deville 86 }.

Ensuite, il y a le système de cas. Les 13 cas définis sont :

Cas		Exemple
Agent	->	Je pars pour le Pérou.
Patient	->	Tu es malade.
Objet	->	Vous devez me renvoyer la balle.
Localisation	->	J'habite à Namur.
Moyen	->	Je dois voter par procuration.
Cause	->	Je voudrais vendre ma trottinette parce que j'ai besoin de sous.
Instrument	->	Le document a été détruit par le feu.
But	->	J'ai besoin d'un visa pour partir à l'étranger.
Bénéficiaire	->	Je vous paie une bière.
Source	->	Je viens des Antilles.
Temps	->	Nous viendrons vous voir demain.
Mesure	->	Je suis mineur.
Condition	->	Faut-il une autorisation si on est étranger ?

Pour une définition précise de ces cas, on consultera { Deville 86 }.

A chaque primitive sont associés des cas. Ceux-ci sont soit obligatoires (Nuclear Case Frame), ou optionnels (Extended Case Frame). Les cas qui ne sont ni obligatoires, ni optionnels sont obligatoirement absents. On a par exemple :

Primitive	Cas obligatoires			Cas optionnels	
	<----------		---------->	<----------	---------->
Ech-Prod.	Agent	Objet	Béneficiaire	Temps	Location
donner	elle	document	me	hier	à Bruxelles

Donc, si on a le prédicat de la phrase, il suffit de voir la primitive de laquelle il est dérivé et d'en déduire les cas qui lui sont attachés.

Sur la base des principes définis dans la grammaire de cas, on peut décrire la structure sémantique profonde d'une phrase au moyen de l'ensemble de règles de la figure III.7. On a ajouté en plus des cas, un ensemble de modalités apportant quelques renseignements supplementaires tels que temps, voix ... de la phrase. On peut alors représenter la structure sémantique d'une phrase sous la forme d'un arbre. Des exemples sont repris en figure III.8.a,b,c

Mais ce qui importe pour notre analyseur, c'est que G. Deville et H. Paulussen ont défini en plus de leur grammaire de cas, défini un ensemble de règles permettant de transformer une représentation syntaxique d'une phrase fournie par les R.N.P. de Pierrel, en une représentation sémantique obéissant aux lois de leur grammaire de cas.

Les règles qu'ils ont définies sont de type condition-action. Les conditions portent sur :

- le prédicat et son environnement syntaxique,
- la phrase nominale et son environnement syntaxique,
- la primitive de laquelle le prédicat est dérivé.

Ces conditions sont évaluées grâce à des informations :

- fonctionnelles syntaxiques (donné par le lexique),
- sur la primitive et sa structure de cas (donné par la grammaire de cas),
- sémantiques (donné par le lexique),

- catégories syntaxiques (données par le lexique).

Ces règles sont reprises en annexe.

Un exemple de transformation d'arbre syntaxique en arbre sémantique est donnée en figure III.9.a,b.

UTTERANCE	:=	MODALITY + PROPOSITION
MODALITY	:=	TENSE/VOICE/MOOD/FORM/MODAL/TYPE
TENSE	:=	present/past/future
VOICE	:=	active/passive
MOOD	:=	infinitive/indicative
TYPE	:=	declarative/ direct interr/ indirect interr
FORM	:=	positive/negative
MODAL	:=	obligation/permission/ability
PROPOSITION	:=	PRIMITIVE + CASE FRAME
PRIMITIVE	:=	VERB/NOMINAL/ADJECTIVE
VERB	:=	dire, donner, ...
NOMINAL	:=	demande, don, ...
ADJECTIVE	:=	belge, divorcé, ...
CASEFRAME	:=	$(CASE + NP/UTTERANCE)^n$
NP	:=	(PREP)+DET(+ADJ)(N)+N+(NP/UTTERANCE)
CASE	:=	agt/pat/obj/ben ...

Figure III.7 : Description de la structure profonde d'une phrase au moyen de règles de structure dans la grammaire de cas de Deville et Paulussen. Tiré de (Deville 86).

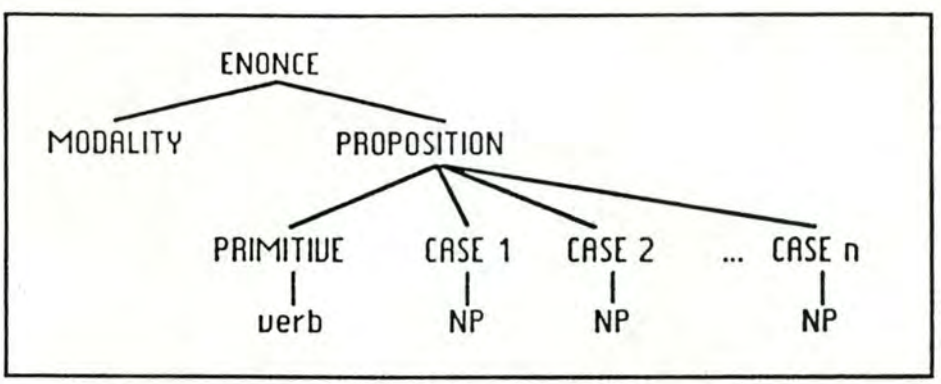
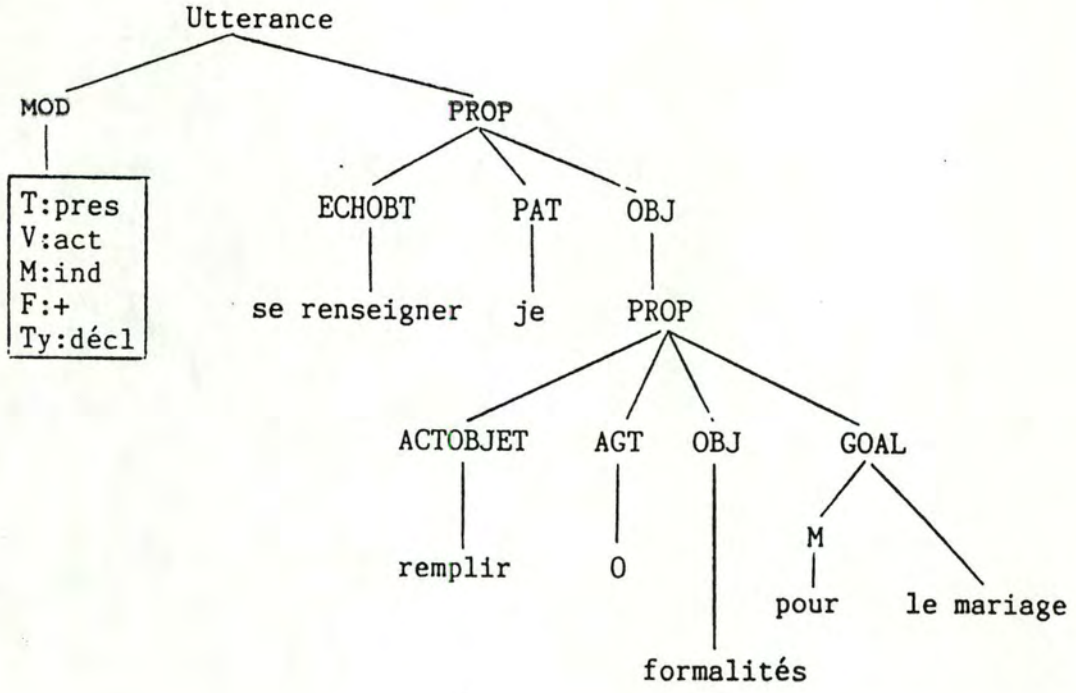


Figure III.8.a : Représentation de la structure sémantique de phrases dans le modèle de la grammaire de cas de Deville et Paulussen. Tiré de (Deville 86).

Je voudrais me renseigner sur les formalités à remplir pour le mariage.



Symbols used :

MOD = modality
T = time : present, past, future
V = voice : active, passive
M = mood : infinitive, indicative
F = form : positive, negative
Ty = type : declarative, direct interrogative, indirect interrogative
Mod = modality : obligation, permission-ability

M in tree refers to case marker

Figure III.8.b : Représentation sémantique de la phrase " Je voudrais me renseigner sur les formalités à remplir pour le mariage ", dans la grammaire de cas de Deville et Paulussen. Tiré de (Deville 86).

Je dois me rendre en Arabie Saoudite. Je crois savoir qu'il faut un certificat de baptême pour y aller. Je n'en ai pas. Je ne suis pas catholique. Je ne suis pas musulman.

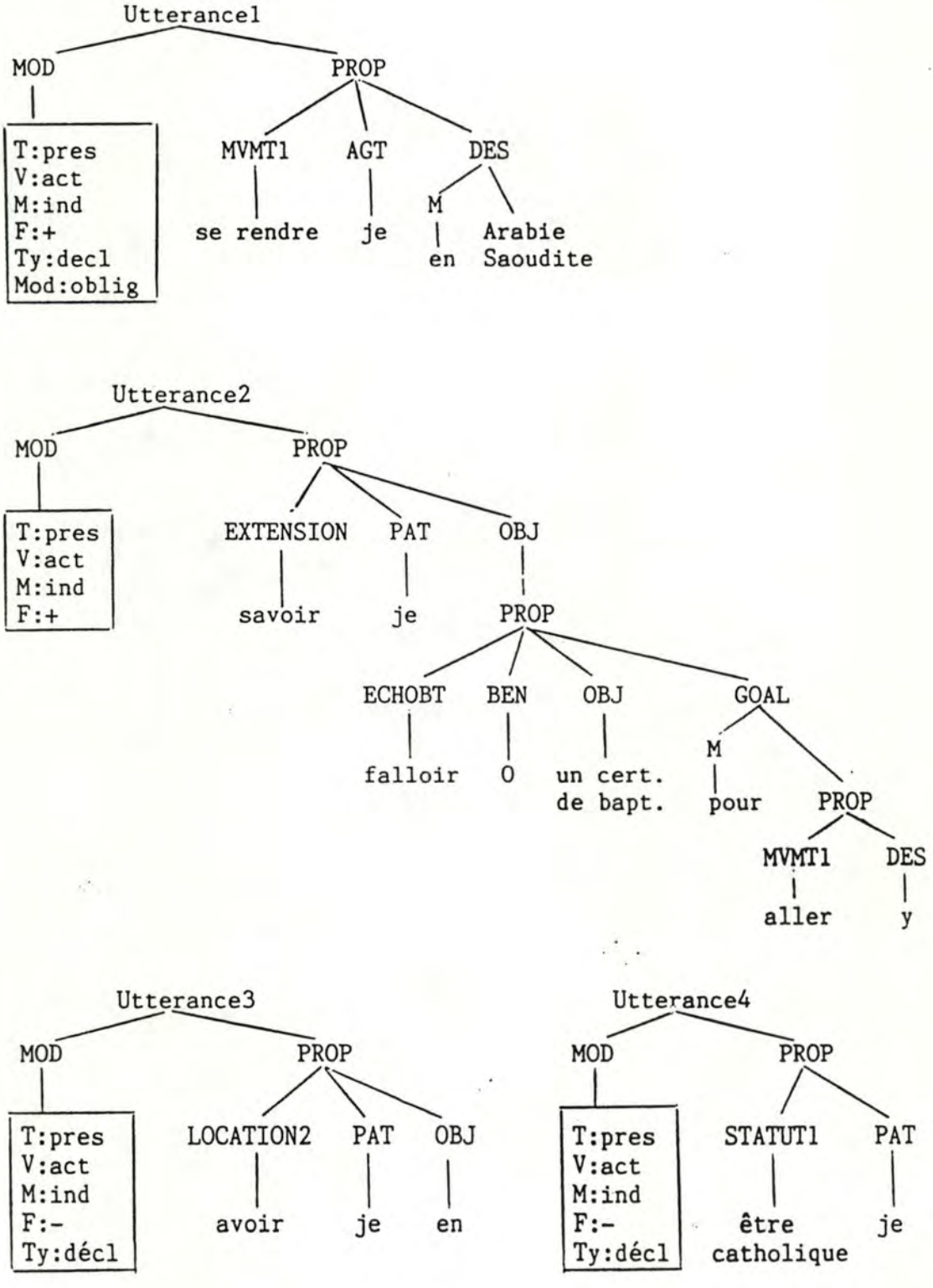


Figure III.8.c : Exemples de représentations sémantiques de phrases dans la grammaire de cas de Deville et Paulussen.
Tiré de (Deville 86).

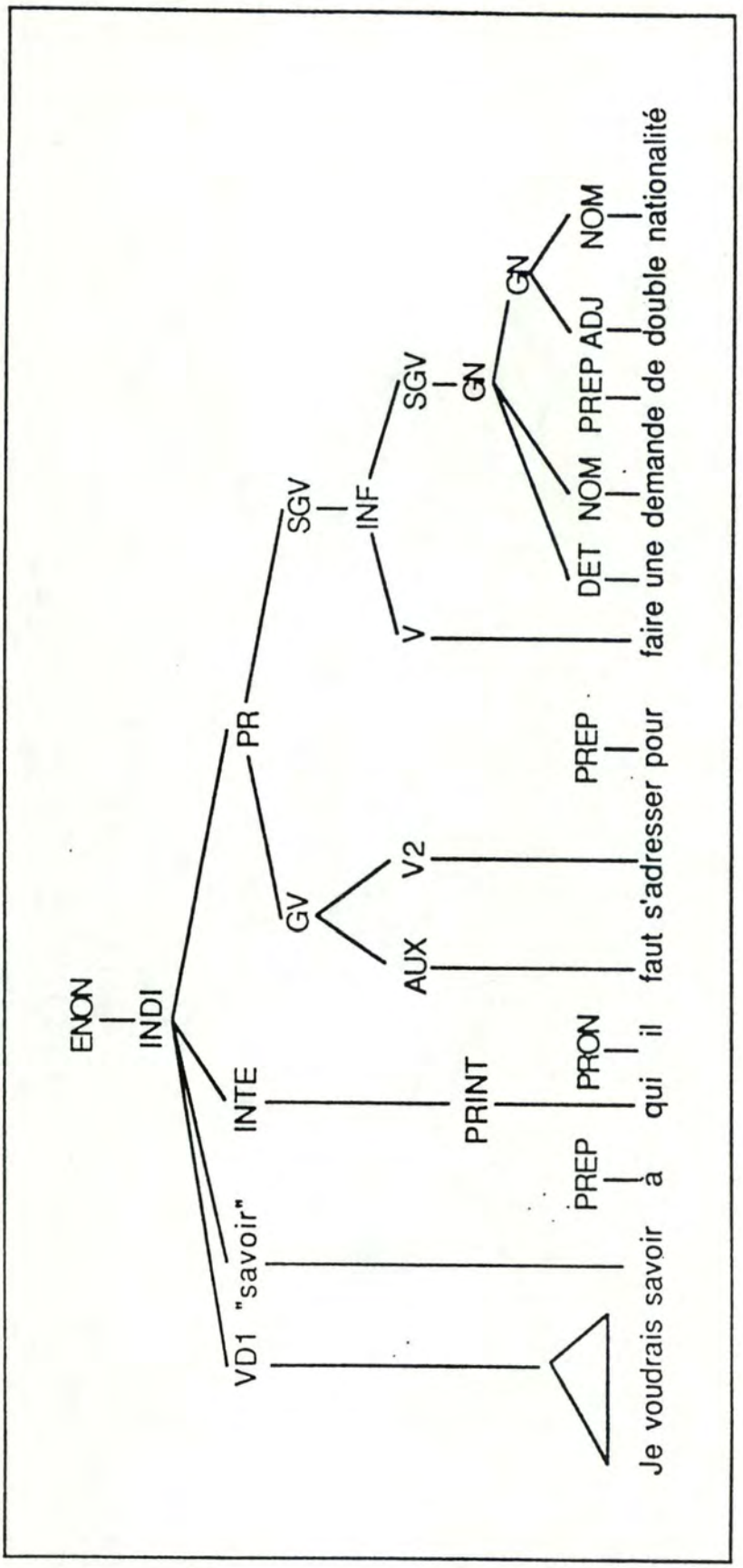


Figure III.9.a : Transformation d'arbre syntaxique en arbre sémantique.
a) Arbre syntaxique.

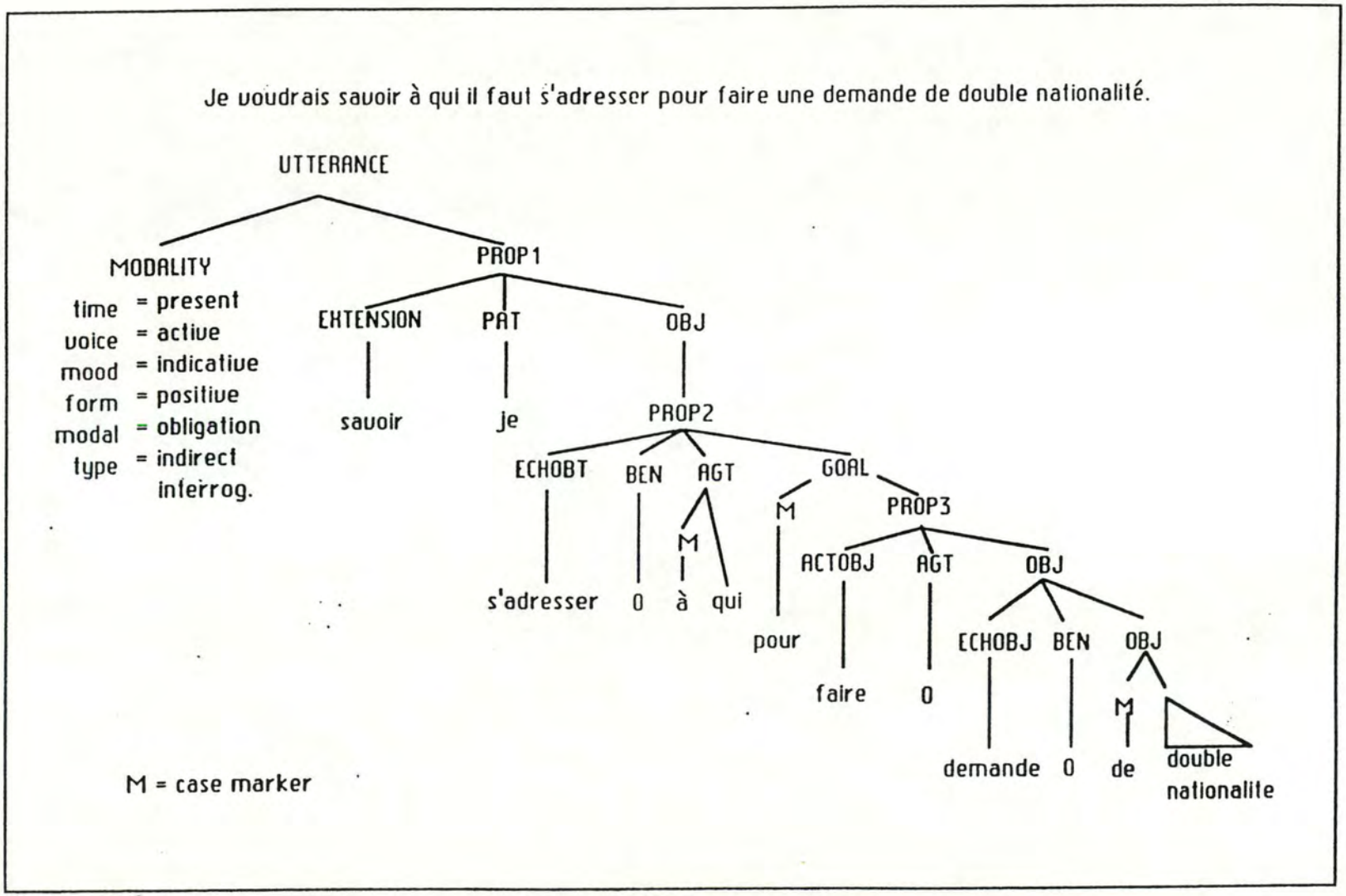


Figure III.9.b : Transformation d'arbre syntaxique en arbre sémantique.
b) Arbre sémantique.

3.4 Les modèles lexicaux et/ou sémantiques.

Ce troisième type de modèles s'appuie essentiellement sur la définition des mots et de leur contexte pour représenter l'ensemble des informations nécessaires à la reconnaissance et à la compréhension de phrases.

Remarquons tout d'abord que parmi ces modèles lexicaux, certains s'appuient sur la définition des transitions possibles entre les mots du langage à reconnaître. Les transitions recouvrent à la fois des informations syntaxiques et sémantiques. Ces modèles permettent une exécution très rapide car les réseaux peuvent être précompilés mais ces réseaux étant spécifiques à chaque application, il faut les réécrire à chaque changement d'application.

Ensuite, on remarque que d'autres modèles lexicaux s'occupent de définir chaque mot en y associant des attributs syntaxiques, sémantiques, ... Dans ces modèles, citons en exemple celui des dépendances conceptuelles de Schank { Schank 75 }. Il utilise un système de cas à partir d'un petit nombre d'actions de base et de relations de ceux-ci avec des ensembles de concepts élémentaires. Il a l'avantage { Pierrel 81 } " de représenter par un même réseau 2 énoncés différents mais de même signification. Par contre, il ne prend pas en compte les informations liées à la position des mots dans la phrase. S'il est bien adapté à l'interprétation, il semble particulièrement mal adapté à un processus prédictif de reconnaissance ".

3.4 Conclusion.

Je viens de présenter un certain nombre de grammaires qui sont employées pour reconstituer les structures syntaxique et sémantique des énoncés. Bien entendu, il en existe d'autres. Il suffit de voir la littérature abondante à ce sujet. J'ai tenté de montrer une évolution pour certaines grammaires. Cela pour montrer que chacune apporte un acquis supplémentaire dont on tient compte dans l'élaboration des systèmes suivants (par exemple : R.T.N., A.T.N., R.N.P). Malheureusement, on ne peut pas intégrer tous les

avantages de toutes les grammaires pour en construire une nouvelle. On peut remarquer que, comme le fait Pierrel dans { Pierrel 81 }, " plus on avance dans les recherches, plus on a tendance à lier la syntaxe et la sémantique. La séparation devient de plus en plus artificielle. Les modèles purement syntaxiques sont essentiellement de type génératif, les modèles sémantiques sont le plus souvent très lourds à mettre en oeuvre et les uns comme les autres semblent mal adaptés à la reconnaissance de langages parlés pseudo-naturels."

J'ai également tenté de mettre en évidence que la séparation entre la grammaire proprement dite et les règles de l'analyseur est importante si on veut pouvoir tranférer un module d'analyse syntaxico-sémantique d'une application vers une autre avec un minimum de changements.

Enfin, si je me suis étendu plus en particulier sur 2 grammaires, à savoir celle des R.N.P. de Pierrel et la grammaire de cas de Deville et Paulussen, c'est simplement parce que ce sont ces 2 grammaires qui seront implémentées dans le système de dialogue auquel nous avons collaboré.

Chapitre 4.

Quelques systèmes déjà réalisés.

Depuis près de 30 années, les chercheurs s'intéressent à la reconnaissance et à la compréhension de la parole par la machine. Un certain nombre de petits projets furent lancés pour défricher le terrain et résoudre les premiers problèmes. Ils s'attachèrent à résoudre les problèmes se situant à un niveau proche du signal. Ce furent principalement des systèmes de reconnaissance de la parole. Ils utilisèrent des langages par mots isolés et d'autres possédant beaucoup de contraintes (peu de mots dans le vocabulaire, faible syntaxe, ...). Mais à la fin des années 60, divers grands projets étaient entrepris pour développer des techniques plus appropriées à cette reconnaissance. Le plus important de ces projets fut sans conteste en 1971, le projet ARPA-SUR (Advanced Research Project Agency- Speech Understanding System), financé par le département américain de la défense des Etats-Unis. Il avait comme objectif de développer des machines capables de "comprendre" des phrases en parole continue avec un vocabulaire de 1000 mots. Ce projet engendra des systèmes tels que HARPY (Lea 80), HEARSAY (Lesser 75), HWIM (Woods 76, Lea 80). Néanmoins, il ne faut pas oublier les autres systèmes développés ailleurs dans le monde. Je citerai en exemple pour la France des systèmes : ESOPE (Mariani 82), KEAL (Quintin 82), et également les systèmes MYRTILLE I et II (Pierrel 81), développés par les chercheurs du CRIN de Nancy.

Bien d'autres systèmes ont été développés, avec plus ou moins de succès. Voyons ce que quelques uns des systèmes les plus connus réalisent et comment ils opèrent.

4.1 Hearsay II:

Produit dans les années 70 dans le cadre du projet ARPA, HEARSAY II est tourné plutôt vers la reconnaissance que vers la compréhension. Il nécessite l'emploi d'une syntaxe très stricte et utilise un vocabulaire de plus de 1000 mots.

C'est un système hétérarchique qui se base sur le modèle "blackboard" pour l'interaction des sources de connaissances. Sa stratégie se fonde sur le principe de la réalisation d'une hypothèse et ensuite sur la mesure de sa validité. Cette stratégie

s'applique à tous les niveaux du système.

Ces sources de connaissances sont activées lorsqu'un certain nombre de conditions sont satisfaites à l'intérieur du composant "blackboard".

Le modèle utilisé dans Hearsay est général et applicable à de nombreux problèmes d'interprétation. Pour plus de détails, on consultera { Lea 80, Erman 80 }.

4.2 HWIM.

Egalement produit sous l'égide du projet ARPA-SUR, ce système réalisé par Bolt Beranek et Newmann (BBN), a l'avantage de n'utiliser qu'une syntaxe très peu contraignante.

C'est un système où la hiérarchie entre les différents niveaux (phonétique, lexical, sémantique et syntaxique) est bien marquée.

Un composant supérieur commande aux composants acoustiques, phonétiques et lexical, la création d'un treillis de mots. Le composant sémantique sélectionne alors une suite de mots de ce treillis sur base de critères sémantiques. Ensuite, le composant syntaxique vérifie la validité de cette "théorie" et la complète au besoin.

L'analyse s'arrête lorsque le composant supérieur estime la théorie suffisamment valable. Pour plus de détails, on consultera { Woods 80 , Lea 80 }.

4.3 MYRTILLE 1.

Premier des systèmes de compréhension de la parole développé au CRIN à Nancy, il utilise un langage de moins de 100 mots avec de plus une syntaxe très restrictive. Son but était de mettre en évidence l'utilité de l'analyse syntaxique dans la reconnaissance de la parole. Il fonctionne pour cela sur la base d'une stratégie descendante (top-down). Le module d'analyse syntaxique émet régulièrement des hypothèses sur les mots à reconnaître. Ces hypothèses se font à partir de la syntaxe déjà reconnue et du contexte. Ensuite, ces hypothèses sont vérifiées au niveau lexical.

Ce système est bien adapté au langage artificiel. Il fut essayé dans le cadre de communication avec un standard téléphonique. Pour plus de renseignements, on consultera (Pierrel 81).

4.4 MYRTILLE 2.

Ce système est la continuation des travaux réalisés sur MYRTILLE 1 . Son objectif est de comprendre un langage pseudo-naturel. Ce langage possède 375 mots. Sa syntaxe contient une grande partie de la grammaire du français courant. La sémantique et la pragmatique sont restreints à un univers particulier. Son but est principalement d'être un outil de travail pour étudier l'importance des différentes analyses et stratégies possibles.

4.4.1 Les informations prises en compte.

MYRTILLE 2 utilise les sources d'informations classiques (acoustiques, phonétiques, phonologiques, prosodiques, lexicales, syntaxiques, sémantiques et pragmatiques) qui sont classées selon qu'il s'agit :

- d'informations liées à la structure du langage (surtout celles syntaxiques mais aussi lexicales, prosodiques et sémantiques). Ce premier type devrait être unique.
- d'informations liées à l'application (surtout lexicales et pragmatiques). Ce type d'information est à redéfinir pour toute application.
- d'informations liées au locuteur (essentiellement phonologiques et prosodiques). Cette classe est également à redéfinir pour chaque classe de locuteurs.

En respectant cette classification, MYRTILLE 2 constitue un système paramétrable à 3 niveaux.

4.4.2 Modèles de représentation des informations linguistiques.

Les informations linguistiques sont représentées par

- les R.N.P.
- un lexique qui contient les définitions des mots du vocabulaire. Ce lexique se compose de deux parties pour tenir compte de la séparation entre l'aspect définitionnel (syntaxe et sémantique) et les aspects propres du langage et à la parole (phonétique et phonologie). Ces deux parties sont elles-mêmes structurées de façon hiérarchique.

4.4.3 Fonctionnement général du système.

Il fonctionne sur base de 2 stratégies conjuguées. Une stratégie bottom-up et une stratégie top-down. Ces stratégies suivent un processus de génération d'hypothèses sur la phrase et de vérification de ces hypothèses. L'analyse d'une phrase se fait à partir des mots les plus sûrement reconnus (les îlots de confiance). Ensuite, cette analyse se poursuit vers la gauche et la droite.

Au départ, l'ensemble des hypothèses correspond à l'ensemble des mots du lexique. Ensuite, la prise en compte des différentes sources d'informations permet de restreindre ces hypothèses uniquement aux mots probables de l'énoncé.

La figure IV. présente la structure générale de MYRTILLE 2. Les principaux modules de ce système sont { Pierrel 81 } :

PARSER : Il utilise la définition syntaxico-sémantique des structures du langage et du contexte correspondant à la partie d'énoncé déjà traitée pour diminuer le nombre d'hypothèses possibles.

SEMAN : Il prend en compte les informations fournies par la définition syntaxico-sémantique des mots du dictionnaire et restreint le nombre des hypothèses en fonction des dépendances conceptuelles acceptées par la partie d'énoncé déjà traitée.

PROPHON : Il restreint et pondère les hypothèses émises au niveau

des mots en fonction des traits prosodiques et de la structure phonétique du contexte à reconnaître.

La RECONNAISSANCE PHONETIQUE correspond à l'étape de test des hypothèses (comparaison de la représentation phonétique de référence avec la partie du treillis en entrée).

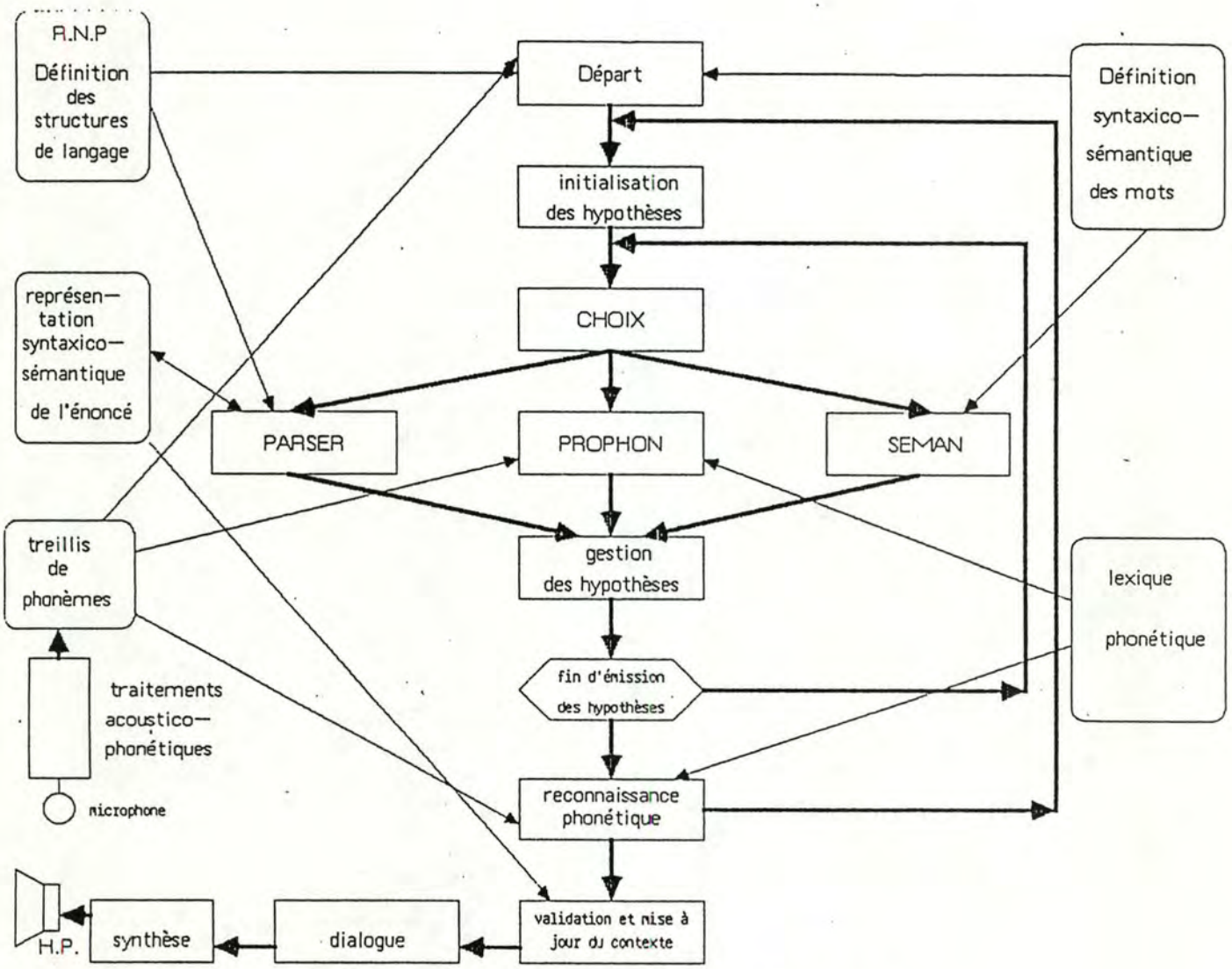


Figure IV.1 : Schéma général du système MYRTILLE 2.

DIALOGUE a pour but d'exploiter les résultats obtenus afin de réaliser l'interprétation sémantique de l'énoncé oral.

VALIDATION a pour fonction de choisir une hypothèse sur base des résultats de la reconnaissance, complète la représentation syntaxico-sémantique de la partie de l'énoncé déjà traitée, et détermine le point de reprise du processus de compréhension en fonction de la stratégie adoptée.

4.4.3 Evaluation.

L'avis même des concepteurs met en évidence,. { Carb 85 } :

" - la non-prise en compte d'informations pragmatiques lors de la phase d'émission des hypothèses.

- le manque d'interaction entre le décodage phonétique et les autres niveaux de traitement.

- l'absence de dialogue permettant une véritable communication homme-machine. "

Ce sont ces quelques remarques concernant les défauts principaux de MYRTILLE 2 qui ont été à l'origine du développement du " système de gestion de dialogue finalisé " en cours de construction au CRIN à Nancy.

4.5 Conclusion.

Nous n'avons ici qu'un minuscule échantillon des systèmes produits jusqu'à présent. Néanmoins, ceux-ci font partie des plus importants et des plus connus. Un simple coup d'oeil sur les performances de ces systèmes permet de nous rendre compte que nous sommes encore loin de la compréhension parfaite de la parole continue par la machine. Myrtille 2 est un système assez récent et s'attache encore à essayer de résoudre des problèmes liés à la nature même de la parole, c'est-à-dire à son aspect linguistique.
De nombreux problèmes ont été résolus grâce à des systèmes

n'utilisant que des langages fortement limités. Ces problèmes étant principalement situés au niveau des connaissances indépendantes de l'application (acoustique, phonétique, phonologiques). Actuellement, la recherche s'oriente donc fortement vers toutes les connaissances linguistiques. On s'oriente doucement vers des systèmes pouvant comprendre la parole continue et gérer réellement un dialogue. En fait, il n'existe pas encore de systèmes possédant un interface oral complet, pratique et hautement fiable. Pour cela, il faudra encore attendre quelques années.

Chapitre 5.

Le système de gestion de dialogues oraux finalisés de Nancy

La nécessité d'un réel dialogue entre l'homme et la machine a déjà été soulignée au chapitre 1. Un échange d'informations satisfaisant entre les interlocuteurs dépend de celui-ci.

C'est cette idée de dialogue qui est à la base du système actuellement en cours à Nancy. Le but est de créer et de mettre au point un véritable système de dialogue homme-machine. C'est ce système en cours de construction qui est présenté dans ce chapitre. Je vais donc commencer par une présentation des objectifs principaux du système. Ensuite, je parlerai des différences essentielles et des améliorations par rapport aux systèmes MYRTILLE 1 et 2 pour montrer que la création du système n'est pas inutile et apporte un nouvel acquis par rapport aux systèmes précédents. Et pour terminer, je décrirai les composants du système et sa stratégie de fonctionnement.

Il me parait évident que la présentation semblera par endroits incomplète et imprécise. Plusieurs raisons peuvent être avancées :

- Le système se base sur les connaissances et l'expérience acquise avec les systèmes MYRTILLE 1 et 2. Il faudrait pour faire une comparaison complète avec ces deux systèmes les connaitre de façon précise. Je ne mentionnerai ici que les principales évolutions par rapport à ces systèmes. Ensuite, ce nouveau système utilise des techniques développées dans les systèmes précédents. Je reprendrai les points importants de ces techniques mais réécrire tout ce qui a déjà été publié auparavant . Je passerai donc outre de certains détails. Il faudra si on désire plus de précisions se référer aux ouvrages concernant les systèmes antérieurs.

- Le système est toujours en cours de développement et un certain nombre de points restent encore à l'étude. Même si les décisions les plus importantes de conception ont été prises, il reste encore un certain nombre de sujets de recherche et divers points risquent encore d'être modifiés par la suite.

- Enfin, je supposerai que le signal acoustique a déjà été traité par un système expert qui a été développé au CRIN. Les détails de ces recherches sont présentés dans { Fohr 85 }.

5.1 Objectifs principaux.

L'équipe de recherche profite de l'expérience acquise avec les projets MYRTILLE 1 et MYRTILLE 2, mais veut avec ce nouveau système passer à une étape supérieure. Il veut dépasser l'analyse de phrases séparées et s'attaquer au véritable problème d'un dialogue en langage naturel. Ce système a pour objectif de permettre son utilisation par le grand public. Il faut donc que l'interface oral soit pratique, efficace et ne demande que peu d'apprentissage. En particulier, il faudrait éviter d'avoir recours à un système de menus peu confortable pour l'utilisateur. Il est donc nécessaire d'avoir un véritable dialogue entre la machine et son interlocuteur, même si au cours de la discussion, c'est la machine qui oriente le dialogue.

Les spécifications principales de ce système sont { Carb 85 } :

" - être capable, dans une certaine mesure, de gérer et de structurer un dialogue oral finalisé.

- disposer d'une expertise dans un domaine relativement limité et simple.

- être capable de comprendre et de satisfaire des requêtes en provenance d'utilisateurs et d'usagers s'exprimant en langage naturel sans restriction ni entraînement préalable. "

Ce système est destiné à être utilisé dans le cadre d'une tâche spécifique. Néanmoins, ces tâches spécifiques peuvent être très variées et atteindre des domaines tels que les domaines commercials, culturels, scientifiques, industriels, ... Par exemple, des demandes de renseignements (-> bases de données), des réservation de places de train, d'avion, des agendas vocaux, ...

5.2. Améliorations par rapport au système précédent.

La première amélioration qui veut être apportée est bien sûr celui de la gestion d'un véritable dialogue entre l'homme et la machine. Il faut aussi que la machine puisse orienter ce dialogue de manière à obtenir toutes les informations nécessaires aux requêtes de l'interlocuteur. Les points de vue sémantique et pragmatique seront donc fortement améliorés. Il y aura prise en compte d'informations sémantiques et pragmatiques dans la gestion du dialogue et la phase d'émission des hypothèses.

Par contre, il y aura peu d'évolution dans les niveaux inférieurs au niveau syntaxique. On se basera sur les systèmes précédents.

Ensuite, d'un point de vue des tâches réalisées par ce nouveau système, une évolution est réalisée par rapport aux 2 systèmes MYRTILLE. Les requêtes qui peuvent être soumises au système sont plus diversifiées :

Dans le cadre des projets MYRTILLE, chaque requête demande en réponse des valeurs spécifiques. Le style de question est toujours du type :

MYRTILLE 1 :

" Bonjour, je voudrais le poste 421, svp. "

MYRTILLE 2 :

" Est-ce que les risques de petites plaques de verglas diminueront demain dans la région de Nancy ? "

Les réponses sont toujours des <u>valeurs</u> ou des <u>identifiants</u> tels que :

- pour des prévisions météorologiques :
  -> température, vitesse du vent, heure, jour, endroit, ..
- dans le cadre de la SNCF.
  -> prix, heure de départ, d'arrivée, destination, réductions, ...

Dans le cadre du système de dialogue oral finalisé, les questions peuvent avoir une portée plus large. Les requêtes peuvent

être du type :

" Je voudrais savoir comment je peux obtenir un visa pour partir aux U.S.A, s.v.p. ? "

Les réponses sont alors des procédures et ne sont plus uniquement des valeurs. Cela compliquera la tâche du module de dialogue.

On se rend compte petit à petit que le module de dialogue jouera un rôle prépondérant dans le système. Notamment au niveau des structures de contrôle.

Examinons maintenant la présentation générale du système et son fonctionnement.

5.2 Les composants du système.

Le système comporte 5 composants. Il sont autonomes et travaillent de manière interactive. Chacun de ces composants correspond à un niveau d'analyse exposé précédemment. Il existe en plus un lexique contenant un ensemble d'informations nécessaires au travail de ces modules.

Le système et les interactions entre les différents composants ou modules sont présentés en figure V.1 .

Les modules sont : APHON, PROSO, LEX, SYN-SEM, DIAL.
Mais que font-ils ?

APHON : C'est le composant qui réalise l'analyse acoustico-phonétique de l'énoncé. Il segmente le signal en phonèmes et tente d'identifier ces différents segments. Ses résultats sont présentés sous la forme d'un treillis phonétique reprenant toutes les solutions probables, avec éventuellement des phonèmes non-identifiés. Ces résultats sont alors transmis au module suivant LEX.

LEX : LEX s'occupe de l'analyse lexicale. Il tente de retrouver les mots et les syntagmes présents dans l'énoncé. Il utilise le

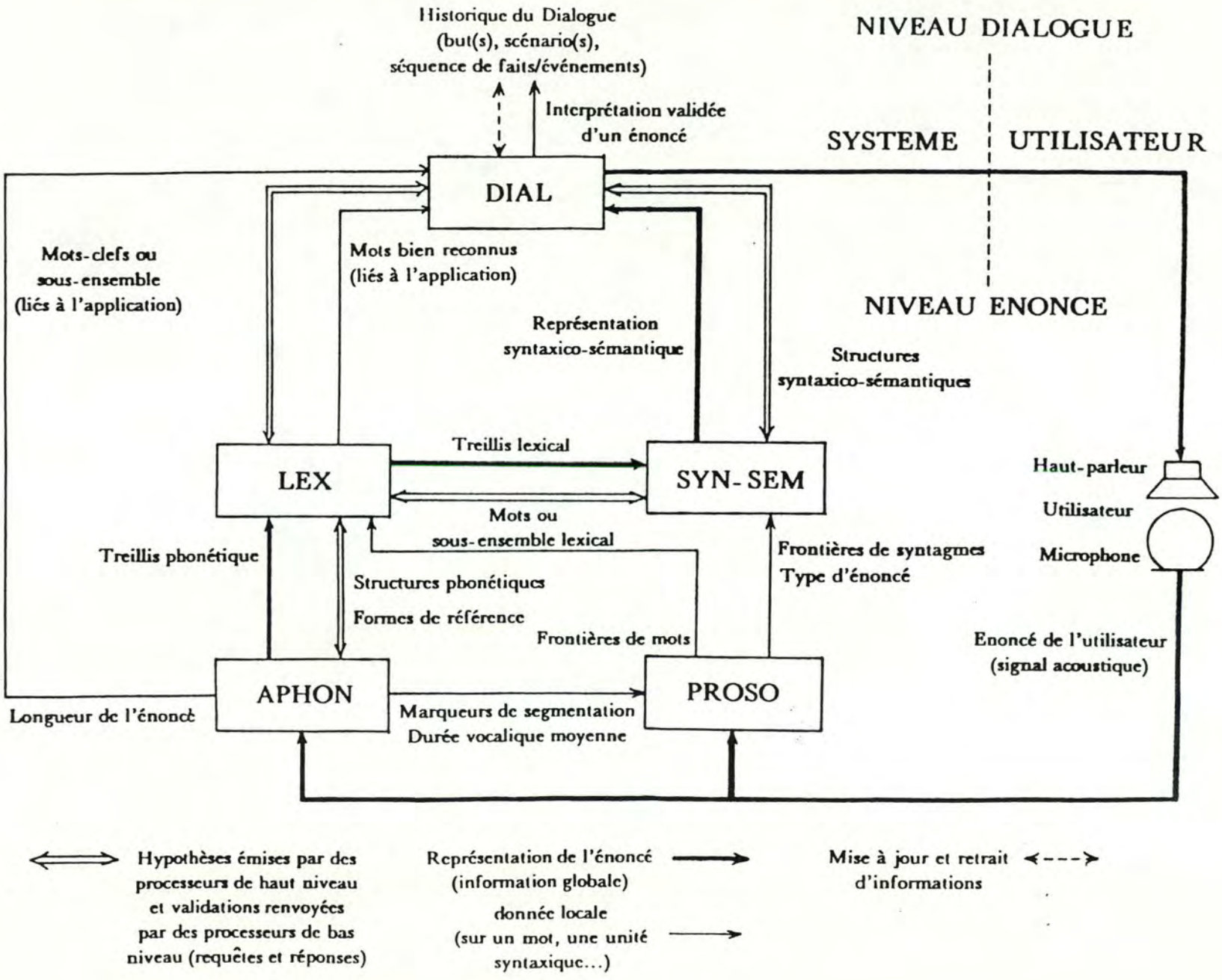


Figure V.1 : Schéma du système de dialogue en construction au CRIN à Nancy.

treillis phonétique qui lui est fourni par le module APHON ainsi qu'un ensemble de représentations phonétiques et phonologiques des mots du vocabulaire (voir point 5.3 Le lexique). Il a, comme expliqué précédemment, deux rôles à jouer. La réalisation de listes de mots hypothèses et la vérification de mots-hypothèses fournis par l'hypothétiseur de mots et le module SYN-SEM. Il produit ses résultats sous la forme d'un treillis lexical qui sera utilisé par le module SYN-SEM.

SYN-SEM : SYN-SEM va essayer de construire une représentation syntaxique et surtout une représentation sémantique de l'énoncé. Il se base sur le treillis lexical fourni par LEX et sur des hypothèses de structure fournies par DIAL. Il utilise pour ce faire, les R.N.P. de J-M. Pierrel pour la construction de la structure syntaxique et la grammaire de cas avec ses règles de transformation à partir des R.N.P. de G. Deville et H. Paulussen pour la construction de la structure sémantique de l'énoncé. Il travaillera en collaboration avec les modules LEX et DIAL.

Le chapitre 6 sera consacré à une étude plus attentive de ce module.

DIAL : Il tient le rôle de commandeur du système. Il interprète l'énoncé en fonction de l'application et du dialogue déjà écoulé. Il se base sur les résultats du module SYN-SEM et sur des données conceptuelles fournies par le lexique. Il traite les diverses interprétations possibles des représentations sémantiques apportées par SYN-SEM (*), en fonction du contexte courant. Il se construit une nouvelle représentation qui est plus adéquate et réalise des inférences, des vérifications de cohérence de ses données, ... pour accroître ses connaissances.
Ensuite, il s'attache à répondre à la requête de l'interlocuteur soit en lui fournissant directement la réponse, soit en lui demandant une confirmation ou un complément d'informations, soit en lui demandant de répéter sa demande. Suivant la réponse, il décidera s'il est nécessaire de relancer toute la procédure de reconnaissance d'un énoncé et d'activer les autres modules. C'est pourquoi il est appelé commandeur du système. Il fournira même le cas échéant des hypothèses de structure à SYN-SEM pour l'aider à analyser l'énoncé de la réponse.

C'est ce module qui gère tout le dialogue. Il doit " raisonner " et cela de façon très complexe. Il se basera sur un ensemble de scénarios préétablis.

(*) En particulier, il traite le problème des références anaphoriques et elliptiques.

- Une ellipse dans un énoncé :

  Une proposition est incomplète ou *elliptique*, lorsque l'usage, le style ou la syntaxe font que l'on exprime pas un ou plusieurs mots auquels l'esprit doit suppléer.

  exemples :

  - énoncé précédent : " Où allez-vous ? "
    énoncé suivant : " A Namur ! "
  - énoncé : " Combien pour un ticket de bus ? "

- Une anaphore dans un énoncé =

  On entend par *anaphore* dans un énoncé donné, tout syntagme contenu dans cet énoncé visant à en remplacer un autre présent dans un énoncé antérieur.

  exemple :

  - énoncé précédent : " Veux-tu une pomme ? "
    énoncé suivant : " J'en ai déjà une. "

PROSO : Il s'occupe de l'analyse prosodique de l'énoncé. Il a 2 rôles principaux.

1°) détecter certaines marques prosodiques indiquant des frontières entre les mots ou les syntagmes.

2°) déterminer la nature de l'énoncé (assertion, question, objection).

Il peut alors aider les modules LEX et SYN-SEM dans leur tâche.

5.3 Le lexique (Pierrel 81, Deville 86).

Pour pouvoir réaliser leur tâche, les différents composants ont besoin de sources de connaissances. Ils possèdent déjà en leur sein l'ensemble des règles définissant leur façon de travailler. Mais en plus, des informations sur les mots du sous-langage utilisé sont nécessaires. Toutes ces informations sont regroupées à

l'intérieur du lexique.

Puisque les différents modules ont accès au lexique, celui-ci est structuré en diverses parties :

- le composant phonologique et phonétique.
- le composant syntaxique et sémantique.
- le composant conceptuel et pragmatique.

5.3.1 Le composant phonologique et phonétique.

Le premier module à utiliser le lexique est le module LEX. Il doit reconnaitre des mots ou des syntagmes à partir du treillis lexical fourni par le module APHON. Il a besoin pour cela des définitions théoriques des mots. Il n'est évidemment pas possible de stocker toutes les prononciations possibles de chaque mot. On a donc adopté quatre types de règles pour trier ces mots et retrouver leurs variations possibles.

- des règles phonologiques : chaque mot est représenté dans sa forme de base. Des règles y sont adjointes pour décrire les variations possibles de ces mots dues aux accords de grammaire (pluriel, féminin, ...), aux coarticulations, ...

- des règles de groupement des mots comportant la même racine ou la même terminaison.
- des règles pour retrouver des mots sur base de caractéristiques acoustico-phonétiques. Par exemple, détection de consonnes plosives : p,t,b,d,k,g ou fricatives : z,v,s,f.
- des règles pour la validation des hypothèses lexicales.

5.3.2 Le composant syntaxique et sémantique.

Utilisé par le module SYN-SEM, à chaque mot sont associés ses catégories syntaxiques (nom, verbe, adjectif, ...) et sémantiques (+/- animé, +/- transitif, ...).

5.3.3 Le composant conceptuel et pragmatique

Le module DIAL a besoin d'informations conceptuelles et pragmatiques. Dans le lexique, de tels concepts existent. Ceux-ci sont représentatifs des objets, des actions, impliqués dans l'univers de l'application. Par exemple, les concepts de nationalité, sexe, personne, ...
Il y a également une définition des relations existant entre ces concepts. Par exemple,

```
père( personne1 , personne2 )
  => ET( enfant( personne2 , personne1 ) , masculin( personne1 ) )
```

devant être lu " La personne1 est le père de la personne2, alors la personne2 est l'enfant de la personne1 et la personne1 est de sexe masculin. "

5.4 Fonctionnement et stratégies du système.

En voulant réaliser un système de compréhension de la parole qui s'apparente fonctionnellement à la compréhension humaine, les créateurs du système ont développé différents modules autonomes. Néanmoins, ceux-ci travaillent en interaction. Chaque composant se charge d'un aspect particulier de l'analyse. Cette autonomie entre les composants permet de garder la possibilité de travailler sur des architectures parallèles et d'améliorer les performances. Les interactions sont nécessaires, car les analyses se font en utilisant des informations venant des autres modules. Les relations qui existent entre les composants sont du type producteur/consommateur.

Le système emploie actuellement 2 stratégies :

- Une analyse de bas en haut (ascendante) à partir du niveau acoustico-phonétique vers les niveaux les plus hauts. Un composant de niveau inférieur fournit les résultats de son analyse au composant de niveau supérieur. Ce dernier réalisera son analyse à partir de ces résultats.

Par exemple, un treillis lexical est fourni par LEX à SYN-SEM qui réalisera l'analyse syntaxico-sémantique sur base de ce treillis.

- Une analyse de haut en bas (descendante) à partir du niveau supérieur DIAL vers les niveaux inférieurs. Un niveau supérieur émet des hypothèses de solution au niveau inférieur que ce dernier se chargera de valider ou d'infirmer.
Par exemple, le module SYN-SEM fournit des hypothèses de mots au module LEX qui se chargera de les vérifier ; le module DIAL fournit des hypothèses de structures au module SYN-SEM en fonction des réponses attendues lors de demandes de confirmation ou de complément d'informations.

Les modules APHON et PROSO sont surtout considérés comme des producteurs d'informations. Ils jouent toutefois le rôle de consommateurs lorsqu'ils évaluent des hypothèses fournies par les modules LEX et PROSO.

5.5 Conclusion.

Ce nouveau système s'attaque à une des difficultés peu abordée dans le passé. Maintenant, il ne s'agit plus de " simplement " comprendre des phrases séparées mais bien de tenter de gérer un véritable dialogue entre l'homme et la machine c'est-à-dire de réaliser un suivi entre les différentes phrases avec un système de questions et de réponses, questions et réponses venant soit de l'interlocuteur, soit de la machine. En fait, il s'agit plus de prouver qu'un tel système est réalisable que de construire un système vraiment opérationnel. C'est en quelque sorte un banc d'essai des systèmes futurs.

Ce système se base sur des connaissances déjà acquises et des résultats produits par les travaux antérieurs. On a vu par exemple que sa structuration est semblable à celle utilisée dans la majeure partie des systèmes de compréhension de la parole se basant sur des informations linguistiques. Il utilise également des méthodes déjà éprouvées dans d'autres systèmes tels que les R.N.P. ayant monré leur efficacité dans le système MYRTILLE 2 et des stratégies de

contrôle qui ont déjà été employées. Néanmoins, ce système n'est pas un simple agglomérat des techniques développées par le passé. Il apporte des connaissances nouvelles, principalement en ce qui concerne le suivi d'un dialogue et l'utilisation pour ce faire de connaissances sémantiques et pragmatiques. Il reste à voir dans quelle mesure les résultats escomptés seront atteints et surtout comment vont se comporter les innovations du système. L'avenir nous le dira.

Chapitre 6.

La composante

Syntaxico-Sémantique.

Dans ce chapitre, je présenterai le module d'analyse syntaxico-sémantique dont le but est de construire, dans la mesure du possible, une représentation syntaxique et une représentation sémantique de l'énoncé.

Je commencerai par rappeller la position de l'analyse syntaxico-sémantique au sein du système. Ensuite, je parlerai des sources de connaissances qu'utilise ce module d'analyse pour atteindre ses objectifs. Je présenterai alors la manière de construire les représentations syntaxique et sémantique de l'énoncé et je détaillerai les diverses interactions de SYN-SEM avec ses voisins. Je continuerai en présentant la structure interne de SYN-SEM et m'arrêterai, en particulier, sur un composant appelé " espace des théories ". Cet espace est destiné à stocker les différentes solutions partielles ou complètes, développées au cours des analyses syntaxique et sémantique de l'énoncé. Je présenterai également la structure du module central de SYN-SEM appelé le superviseur et qui a pour fonction de décider de la tâche à effectuer à un instant donné. J'en arriverai alors à une partie plus technique en présentant les différentes structures de données traitées par l'analyseur syntaxico-sémantique et je terminerai par la présentation d'un ensemble de fonctions d'accès à l'espace de théories. Ces fonctions sont destinées à permettre l'emploi de cet espace de théories. La définition et la programmation de ces fonctions constitue essentiellement mon apport personnel au système.

6.1 Situation de l'analyse syntaxico-sémantique.

Diverses analyses d'un énoncé sont effectuées avant d'arriver à comprendre un énoncé (voir chapitres 2, 5). Le but de l'analyse syntaxico-sémantique est de construire une représentation

syntaxique et une représentation sémantique d'un énoncé qui soient suffisantes pour permettre la compréhension de cet énoncé.

L'analyse syntaxico-sémantique est réalisée par le module SYN-SEM et prend place entre l'analyse lexicale réalisée par le module LEX et l'analyse sémantico-pragmatique réalisée par le module DIAL.

Le système utilise 2 stratégies de fonctionnement : une stratégie ascendante dans laquelle les analyses de l'énoncé sont réalisées à partir du niveau acoustique et en poursuivant vers les niveaux supérieurs, et une stratégie descendante dans laquelle les analyses sont réalisées en commençant par celles de plus haut niveau (sémantiques et pragmatiques) et en continuant vers les niveaux les plus bas.

6.2 Sources de connaissances utilisées par SYN-SEM.

Les différentes sources de connaissances utilisées par SYN-SEM sont :

- les R.N.P. de J-M Pierrel. Ces R.N.P. décrivent la structure syntaxique du langage et contiennent des règles permettant de construire une représentation syntaxique de l'énoncé.

- la grammaire de cas de G. Deville et H. Paulussen qui décrit la structure sémantique possible des énoncés (relations qui peuvent exister d'un point de vue du sens entre les différents éléments d'une phrase). A cette grammaire est associée un ensemble de règles permettant de transformer une représentation syntaxique respectant les règles des R.N.P. en une représentation sémantique satisfaisant aux règles de la grammaire de cas de Deville et Paulussen.

Ces 2 grammaires sont présentées dans le chapitre 3.

- le lexique, qui contient des informations syntaxiques et sémantiques au sujet des mots du vocabulaire. Ce lexique est présenté au chapitre 5.

- des informations lexicales apportées par le module LEX.

- des informations de structures possibles de l'énoncé fournies par le module DIAL.

6.3 Construction des représentations syntaxiques et sémantiques. Interactions de SYN-SEM avec ses modules voisins.

6.3.1 Fonctionnement général du système.

Il faut se souvenir que c'est le module DIAL qui est le noyau du système car il tient le rôle du meneur de dialogue. C'est lui qui connaît et détermine les différentes phases du dialogue. Il décide donc du moment où il faut activer les autres composantes pour effectuer la reconnaissance d'un énoncé, et également du moment où l'analyse est suffisante.

Lorsque le module DIAL décide de commencer la reconnaissance d'un énoncé, il peut

- soit déclencher une analyse descendante. Dans ce cas, il active la composante SYN-SEM et, en même temps, il lui fournit des hypothèses sur la structure syntaxico-sémantique possible de l'énoncé.

- soit déclencher une analyse ascendante et dans ce cas, DIAL active la composante APHON qui est chargée de l'analyse acoustico-phonétique du signal. APHON transmettra ses résultats au module LEX et ensuite ce sera LEX qui transmettra ses résultats (des hypothèses de mots) au module SYN-SEM.

- soit déclencher simultanément ces deux types d'analyse.

En règle générale, le système utilise en même temps les deux types d'analyse et l'architecture modulaire de celui-ci permet aux différents modules de travailler en parallèle.

6.3.2 Construction des représentations syntaxiques et sémantiques.

Les représentations syntaxiques et sémantiques se présentent sous la forme de structures arborescentes satisfaisant aux règles des R.N.P. et de la grammaire de cas de Deville et Paulussen.

Dans le cadre d'une stratégie ascendante, la construction des représentations syntaxique et sémantique débute avec l'arrivée des hypothèses lexicale de LEX. La première représentation est donc constituée de mots isolés. Il s'agira ensuite de reconstruire les arbres syntaxique et sémantique. SYN-SEM ajoutera alors différents noeuds sur base de ces mots jusqu'à reconstruire ces arbres .

Dans le cadre d'une stratégie descendante, ce sont des hypothèses de structure venant de DIAL qui constituent les premières représentations syntaxiques et sémantiques de la phrase. Ces représentations sont ensuite développées jusqu'à retrouver les mots de la phrase.

Voyons comment reconstruire un arbre.

Reconstruire un arbre, c'est ajouter progressivement des noeuds à cet arbre. Rappelons d'abord qu'il existe trois types de noeuds sur un arbre. Il y a des noeuds correspondant aux terminaux et aux non-terminaux de la grammaire, et des noeuds "réalisation" de ces terminaux. En effet, il faut faire la différence entre un noeud terminal de la grammaire, par exemple : NOM, VERBE, ... et un noeud réalisation de ce terminal, par exemple : AUTORISATION, MANGER, ...

Les règles des grammaires et les informations lexicales précisent les noeuds qui peuvent être ajoutés aux différents endroits de l'arbre pour respecter la définition du langage de l'application. Les règles de grammaire tiennent compte des noeuds terminaux et des noeuds non-terminaux qui peuvent être ajoutés. Les informations lexicales permettent de placer de nouveaux noeuds réalisation.

Tant que SYN-SEM s'occupe de placer de nouveaux noeuds non-terminaux sur les arbres, il n'a besoin que des règles contenues dans les banques de données syntaxiques et sémantiques. Mais lorqu'il désire placer un noeud terminal, il a besoin de valider ce noeud par un noeud réalisation. Cette validation se fait de la manière suivante. Nous savons que SYN-SEM travaille en parallèle avec LEX. Ce dernier apporte régulièrement à SYN-SEM des hypothèses de mots qu'il croit avoir reconnus à un endroit de l'énoncé. Alors, soit le mot dont SYN-SEM a besoin pour valider le noeud terminal a déjà été reconnu par LEX à cet endroit de l'énoncé, soit il ne l'est pas. S'il l'est, pas de problèmes. Mais dans le second cas, SYN-SEM proposera alors à LEX une hypothèse de mot et LEX sera chargé de vérifier cette hypothèse. Suivant le diagnostic de LEX, SYN-SEM poursuivra son analyse soit en ajoutant ce noeud réalisation sur l'arbre, soit en essayant une autre solution pour le développement de cet arbre.

L'adjonction de noeuds aux arbres se fait suivant les règles définies dans les grammaires. Cependant, en général, plusieurs de ces règles sont applicables c'est-à-dire qu'il existe plusieurs manières de développer une représentation. On peut alors choisir soit de développer une seule représentation complètement (jusqu'à reconstruire un arbre complet), soit de développer simultanément toutes les représentations possibles, soit de n'en développer que quelques-unes (les quelques meilleures, les plus plausibles). C'est cette troisième solution qui a été adoptée.

" Développer les quelques meilleures représentations simultanément " signifie :

- Choisir une première représentation parmi les meilleures.
- Lui ajouter un ou plusieurs noeuds jusqu'à être obligé d'émettre une hypothèse avant de poursuivre le développement de cette représentation. Cette représentation est alors mise en attente jusqu'à la réception du diagnostic de validité de cette hypothèse.
- Ensuite, on choisit alors une autre représentation parmi les meilleures et on la développe également jusqu'à émettre une hypothèse. De nouveau, on abandonne cette représentation

temporairement et

- on recommence en choisissant parmi les meilleures une nouvelle représentation à développer (éventuellement une représentation dont l'hypothèse à été évaluée).

L'abandon temporaire d'une représentation nous obligera à associer à celle-ci, divers attributs qui permettront de reprendre son développement. Un attribut serait par exemple la dernière des règles qui a été utilisée lors le développement de cette représentation.

On peut signaler que lorsque les deux types d'analyse sont réalisées simultanément, des représentations syntaxiques et sémantiques provenant d'une analyse ascendante cotoieront des représentations syntaxiques et sémantiques créees par l'analyse descendante. Il est donc possible que deux représentations de type différent fusionnent pour en former une nouvelle.

6.3.3 Communication synchrone et asynchrone.

Nous connaissons déjà plusieurs instants où SYN-SEM communique avec LEX et DIAL.

La première de ces communications se produit au moment de l'activation de SYN-SEM par le module DIAL. Ce dernier fournit, en même temps, à SYN-SEM des hypothèses de structure. La communication de ces hypothèses à SYN-SEM se fait donc de manière synchrone du point de vue de SYN-SEM.

Nous savons également que les différents modules travaillent simultanément. En particulier, le module lexical peut faire de la reconnaissance de mots pendant que le module syntaxico-sémantique est en train de développer ses propres théories. Ces deux modules travaillent indépendamment. SYN-SEM ne peut donc pas connaître le moment précis où le module LEX aura reconnu un mot. Ce mot, considéré comme hypothèse lexicale va être transmis à SYN-SEM de manière asynchrone du point de vue de ce dernier.

Un second moment de communication entre LEX et SYN-SEM est

celui où, étant en train de développer une représentation, SYN-SEM en arrive à demander la validation d'un mot à LEX. Cette hypothèse-mot est communiquée à LEX de façon synchrone du point de vue de SYN-SEM.

En attendant la vérification de son hypothèse-mot, SYN-SEM peut, par exemple, travailler sur une autre représentation. Mais même s'il ne fait rien, il ne pourra jamais savoir le moment précis où LEX aura terminé d'examiner cette hypothèse. Le diagnostic (degré de validité de l'hypothèse) de cette hypothèse sera donc renvoyé à SYN-SEM de façon asynchrone du point de vue de SYN-SEM.

Enfin, SYN-SEM communiquera à DIAL les hypothèses qu'il a développées.

On se rend compte que du point de vue de SYN-SEM, la communication se fait toujours de manière synchrone avec DIAL mais par rapport à LEX, cette communication est réalisée par moment de manière synchrone et à d'autres moments de manière asynchrone.

6.4 Architecture interne du module SYN-SEM.

L'architecture interne du module SYN-SEM est présentée à la figure VI.1.

On y remarque la présence :

- d'une banque de données syntaxiques contenant la définition des Réseaux à Noeuds Procéduraux utilisés pour l'analyse syntaxique.

- d'une banque de données sémantiques contenant la grammaire des cas de Deville et Paulussen et les règles de transformations d'hypothèses syntaxiques en hypothèses sémantiques utilisées lors de l'analyse.

- d'un compilateur permettant à un utilisateur de modifier la définition des R.N.P.. Ce compilateur transforme une représentation externe des R.N.P. en une représentation

interne de celle-ci.

- d'un compilateur permettant à un utilisateur de modifier la définition de la grammaire de cas. Ce compilateur transforme une représentation externe de la grammaire de cas en une représentation interne de celle-ci.

- d'interfaces entre les compilateurs et les utilisateurs qui permettent à ces derniers de modifier les grammaires en fonction de l'application.

- d'un module de raisonnement. C'est un module qui a la tâche de décider des actions à réaliser à un certain moment pour mener à bien les analyses syntaxique et sémantique.

- d'un interface d'accès aux informations syntaxiques à partir du module de raisonnement . Il contient un certain nombre de fonctions destinées à permettre une utilisation efficace des connaissances syntaxiques par le module de raisonnement.

- d'un interface d'accès aux informations sémantiques à partir du module de raisonnement. Il contient un certain nombre de fonctions destinées à permettre une utilisation efficace des connaissances sémantiques par le module de raisonnement.

- d'une zone de stockage des résultats partiels ou terminaux des analyses syntaxiques et sémantiques. Nous appelons ces résultats des " théories " (bien que celles-ci n'aient aucun rapport avec des théories scientifiques) et cette zone " l'espace ou la base de théories ". Cet espace et son contenu exact sont présentés dans le point 6.5.

- d'un interface d'accès à l'espace des théories à partir du module de raisonnement. Il contient un certain nombre de fonctions destinées à permettre une utilisation efficace de l'espace de théories par le composant de raisonnement.

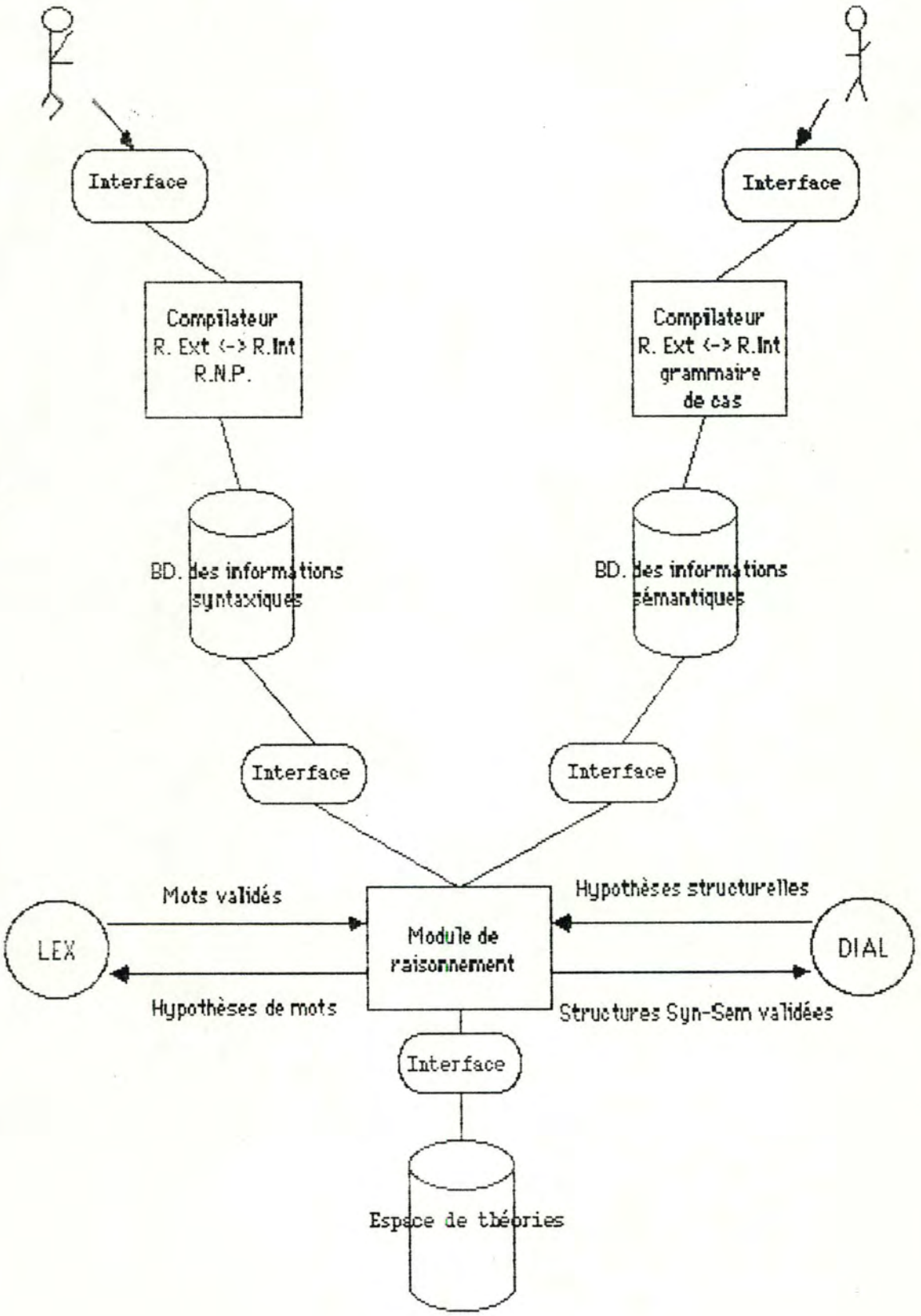


Figure VI.1 : Schéma interne du module SYN-SEM.

6.5 L'espace des théories.

D'abord, plus précisément, qu'est-ce qu'une théorie?

A un moment quelconque de l'analyse syntaxico-sémantique, il y a un certain nombre de représentations incomplètes et peut-être même un certain nombre de représentations complètes déjà créées. Une théorie est un résultat partiel ou total de(s) l'analyse sémantico-syntaxique de l'énoncé qui permet d'émettre des hypothèses (de type mot vers LEX ou de type structure vers DIAL). Elle est composée soit d'une représentation sémantique de l'énoncé et/ou d'une représentation syntaxique de l'énoncé. Ces représentations sont accompagnées d'un ensemble d'informations qui autorisent, par la suite, la reprise du développement des théories dont les représentations sont incomplètes.

Une théorie peut ne comporter qu'une représentation sémantique de l'énoncé lorsque la représentation syntaxique s'avère impossible à construire. En effet, en général, les diverses théories contiennent une représentation syntaxique et une représentation sémantique de l'énoncé. Malheureusement, il est possible qu'à un certain moment, l'analyseur ne soit plus capable de poursuivre le développement d'une représentation encore incomplète (la phrase peut être mal construite, une erreur s'être produite précédemment dans une analyse, ...). Cependant, l'abandon de la construction de cette représentation ne doit pas empêcher la poursuite du développement de la seconde. On trouvera alors parmi les théories celles qui contiennent deux représentations et celles qui n'en contiennent plus qu'une.

Dans notre problème de compréhension de la parole, des théories purement syntaxiques (qui ne contiendraient que des représentations syntaxiques des phrases) n'ont pas un grand intérêt, car ce sont uniquement les structures sémantiques qui sont destinées au module DIAL. Les structures syntaxiques sont intéressantes uniquement parce qu'elles sont sources de beaucoup de contraintes et permettent ainsi de limiter le nombre de théories sémantiques possibles de l'énoncé. Nous ne garderons donc que les théories qui contiennent soit une représentation sémantique, soit une représentation sémantique et une représentation syntaxique.

Une théorie ascendante contiendra des représentations de l'énoncé créées à partir d'une analyse ascendante. De même, une théorie descendante contiendra des représentations de l'énoncé créées à partir d'une analyse descendante.

La fonction de l'espace des théories est de stocker les différentes théories. Cet espace constitue une ressource critique. En effet, de nombreuses fonctions de SYN-SEM doivent accéder à cet espace pour diverses raisons et à des instants imprévisibles. Par exemple :

- A partir des hypothèses de structures émises par DIAL, SYN-SEM construit des théories syntaxico-sémantiques qu'elle doit insérer dans l'espace des théories.

- A partir des hypothèses lexicales émises par LEX, SYN-SEM construit des théories syntaxico-sémantiques qu'elle doit insérer dans l'espace des théories.

- A partir des hypothèses syntaxico-sémantiques de l'espace des théories, SYN-SEM construit de nouvelles théories syntaxico-sémantiques qui doivent à leur tour être insérées dans l'espace des théories.

Vu l'importance de cette ressource, il faut en définir l'accès. On ne peut pas déterminer à priori quelles opérations seront faites à un moment précis. Le mécanisme d'accès peut alors être de deux types: soit un mécanisme d'interruption, soit un mécanisme de scrutation. C'est le second qui nous a paru le plus aisé à mettre en oeuvre.

6.6 Raisonnement du superviseur.

Le module SYN-SEM construit progressivement les représentations syntaxique et sémantique de l'énoncé. Il développe en même temps plusieurs représentations correspondant à plusieurs solutions qu'il estime être possibles. Il commence par créer des représentations très incomplètes de la phrase. Puis il construit des représentations-filles des représentations déjà créées en leur

ajoutant de nouveaux éléments et cela jusqu'à obtenir des représentations de la phrase qui sont estimées suffisantes pour être utilisées par le module DIAL.

Il faut donc un superviseur qui décide des actions à effectuer à un moment précis. Ce superviseur est contenu dans le module de raisonnement. La structure de ce superviseur est reprise de { Mousel 87 } en figure VI.2.

6.7 Implémentation.

L'implémentation du système à été réalisée sur un VAX 785 utilisant un operating system UNIX (version BSD 4.2). Le langage utilisé est le langage C pour des raisons de portabilité. L'implémentation des R.N.P. ainsi que ses fonctions d'accès a été réalisée par P. Mousel { Mousel 86 }, celle de la grammaire de cas de Deville et Paulussen ainsi que ses fonctions d'accès sera également réalisé par P. Mousel. L'implémentation de la base de théories ou plutôt la réalisation de ses fonctions d'accès était le but de ce mémoire.

Structure du Superviseur de la composante SYN-SEM

```
Toujours
--------
   Début
   -----

   Tant que pas d'hypothèses dialogue, de mots validés ou
   --------
                d'hypothèses lexicales
       Début
       -----
       Attendre

       Fin
       ---
   Pour toute hypothèse dialogue
   ----------
       Début
       -----
       Transformer l'hypothèse dialogue en théorie
         syntaxico-sémantique
       Intégrer la théorie ainsi obtenue dans l'espace des
         théories

       Fin
       ---
   Pour tout mot validé
   ---------
       Début
       -----
       Construire la théorie syntaxico-sémantique fille
         à partir du mot validé et de la théorie
         syntaxico-sémantique qui a permis de l'obtenir
       Intégrer la théorie ainsi obtenue dans l'espace des
         théories

       Fin
       ---
   Pour toute hypothèse lexicale
   ----------
       Début
       -----
       Transformer l'hypothèse lexicale en théorie
         syntaxico-sémantique
       Intégrer la théorie ainsi obtenue dans l'espace des
         théories

       Fin
```

Figure VI.2 : Structure du superviseur. (Tiré de Mousel 87)

```
    ---
Pour toute théorie ascendante
----------
    Pour toute théorie descendante
    ----------
        Début
        -----
        Essayer de raccrocher la théorie ascendante à la
          théorie descendante
        Intégrer la théorie ainsi obtenue dans l'espace
          des théories
        Fin
        ---

    Sélectionner des théories dans l'espace des théories

    Pour toute théorie sélectionnée
    ----------
        Début
        -----
        Activer l'analyseur syntaxico-sémantique déterminant les
          hypothèses mots

        Fin
        ---
Fin
---
```

Autour de ce noyau viennent se greffer les différentes fonctions syntaxico-sémantiques proprement dites qui sont:

- Transformation d'hypothèses lexicales en théories syntaxico-sémantiques.
- Transformation d'hypothèses structurelles en théories syntaxico-sémantiques.
- Construction de théories filles à partir de théories syntaxico-sémantiques et de mots validés.
- Intégration de théories dans l'espace des théories.
- Sélection de théories dans l'espace des théories.
- Analyseurs syntaxico-sémantiques.

Figure VI.2 : Structure du superviseur. (Tiré de Mousel 87)

Par la suite, je décrirai l'espace des théories, appelé aussi " base de théories ", ce qu'il va contenir et les fonctions d'accès à cette base.

6.8 La structure de l'espace des théories.

6.8.1 Les différentes théories.

Nous savons que le système réalise deux types d'analyse : une analyse ascendante et une analyse descendante. Nous savons également qu'une théorie peut contenir une ou deux représentations.

Les opérations qui peuvent être réalisées pour développer une théorie ne sont pas identiques suivant que cette théorie est de type ascendant, descendant et possède une ou deux représentations. Nous grouperons dès lors les théories en quatre ensembles distincts et disjoints, un ensemble par type de théorie.

- des théories syntaxico-sémantiques descendantes.
- des théories syntaxico-sémantiques ascendantes.
- des théories purement sémantiques descendantes.
- des théories purement sémantiques ascendantes.

6.8.2 Structure abstraite de l'espace des théories.

L'espace des théories est composé de théories. Une théorie est identifiée par un numéro. Elle comprend un arbre sémantique et/ou un arbre syntaxique.
Un arbre syntaxique se compose de noeuds syntaxiques.
Un arbre sémantique se compose de noeuds sémantiques.
Un noeud syntaxique est identifié par son type, son symbole (sa valeur) et ses liaisons avec son noeud-père et ses noeuds-fils.
Un noeud sémantique correspond à l'énoncé de toute une proposition de la phrase. Il se compose d'une liste de modalités; du type, du nom et de la valeur de la réalisation de la primitive; d'une liste de cas.
Une liste de cas se compose de cas.

Un cas est identifié par son type, son symbole et sa réalisation. La réalisation d'un cas peut se faire soit par un énoncé complet de proposition, soit par un groupe nominal.
Un groupe nominal est composé d'une préposition, d'un déterminant, d'une liste d'adjectifs et d'un nom.
Une liste de modalités se compose de modalités.
Une modalité est identifiée par son nom et par sa valeur.

Ces différentes notions ont été présentées au chapitre 3, dans la partie décrivant la grammaire de cas de Deville et Paulussen.

6.8.3 Structure concrète de l'espace des théories.

La structure décrite dans le paragraphe précédent doit être implémentée de façon à répondre à certaines contraintes de traitement. En effet, nous envisageons des analyses ascendantes, descendantes, de la gauche vers la droite ainsi que du milieu vers les côtés. Il faudra donc définir une structure concrète, éventuellement redondante, mais permettant de retrouver rapidement les informations nécessaires aux différents traitements envisagés.

Lors de l'implémentation, nous décomposerons une théorie en ses différents composants qui viennent d'être présentés dans la représentation abstraite d'une théorie. Cela pour des raisons de facilité et parce qu'il est très difficile, si pas impossible, de trouver une structure unique pouvant contenir toutes les informations concernant une théorie. Les différentes structures utilisées pour représenter une théorie comprendront :

- des informations permettant d'identifier ces structures.
- des informations relatives à ces structures et permettant de développer les théories.

Décrivons maintenant la structure complète des théories.

6.8.3.1 Les théories Syntaxico-Sémantiques Descendantes (SSD).

- A chaque théorie SSD est associé un numéro identifiant.

- A chaque théorie SSD sont associées une structure d'arbre syntaxique et une structure d'arbre sémantique.

- En général, chaque théorie SSD est fille d'une ou de plusieurs théories syntaxico-sémantiques (ascendantes ou descendantes) mères (Voir figure VI.3) :

  - à l'exception des théories SSD initiales obtenues à partir des hypothèses émises par la composante DIAL; mais à part celles-ci :

  - chaque théorie SSD est de toute façon fille d'une théorie SSD, mais peut de plus être fille d'une théorie syntaxico-sémantique ascendante dans le cas d'un rattachement d'une théorie syntaxico-sémantique ascendante à une théorie syntaxico-sémantique descendante.

- En règle générale, chaque théorie SSD est mère d'une ou de plusieurs théories SSD filles ou d'une théorie purement sémantique descendante fille (pour les contraintes de parenté, voir les théories sémantiques descendantes), à l'exception de :

  - des théories englobant tout l'énoncé,

  - des théories abandonnées pour cause de score trop faible,

  - des théories en cours d'étude.

  Chaque théorie SSD fille a été obtenue en opérant une ou plusieurs modifications dans les arbres syntaxique(s) et/ou sémantique(s) de la (des) theorie(s) SSD mère(s) :

  - soit parce qu'un ou plusieurs noeuds ont été ajoutés à l'arbre syntaxique de la théorie SSD mère et/ou parce qu'un ou plusieurs ajouts ont été effectués sur l'arbre sémantique de la théorie SSD mère. Ces modifications permettent au module SYN-SEM d'émettre au moins une

hypothèse vers LEX ou DIAL.

- soit parce que la théorie SSA mère a été rattachée à la théorie SSD mère.

L'indication de ces relations mères-filles permet de retrouver les différentes étapes du développement d'une théorie en observant les modifications qui sont réalisées lorsqu'on passe de la théorie-mère à la théorie-fille.

- A chaque théorie SSD est associé un type. En l'occurence, le type théorie SSD. (THSSD)

- A chaque théorie SSD est associé un score définissant son degré de vraisemblance. Ce score permet d'évaluer s'il vaut la peine de continuer à développer cette théorie.

Ces deux derniers attributs sont utilisés spécialement lors du développement de cette théorie.

6.8.3.2 Les théories Syntaxico-Sémantiques Ascendantes (SSA).

- A chaque théorie SSA est associé un numéro identifiant.

- A chaque théorie SSA sont associées une structure d'arbre syntaxique et une structure d'arbre sémantique.

- En général, chaque théorie SSA est fille d'une ou de plusieurs théories SSA mères (Voir figure VI.3) :

  - à l'exception des théories SSA initiales obtenues à partir des hypothèses émises par la composante LEX; mais à part celles-ci :

  - chaque théorie SSA est de toute façon fille d'une théorie SSA, mais peut de plus être fille d'une deuxième théorie SSA dans le cas d'un rattachement de cette dernière à la première.

- En règle générale, chaque théorie SSA est mère d'une ou de plusieurs théories syntaxico-sémantiques (ascendantes ou descendantes) filles ou d'une théorie purement sémantique ascendante fille (pour les contraintes de parenté, voir les théories syntaxico-sémantiques descendantes ainsi que les théories sémantiques ascendantes), à l'exception :

  - des théories englobant tout l'énoncé,

  - des théories abandonnées pour cause de score trop faible

  - des théories en cours d'étude.

  Chaque théorie SSA fille a été obtenue en opérant une ou plusieurs modifications dans les arbres syntaxique(s) et/ou sémantique(s) de la (des) theorie(s) SSA mère(s) :

  - soit parce qu'un ou plusieurs noeuds ont été ajoutés à l'arbre syntaxique de la théorie SSD mère et/ou parce qu'un ou plusieurs ajouts ont été effectués sur l'arbre sémantique de la théorie SSD mère. Ces modifications permettent au module SYN-SEM d'émettre au moins une hypothèse vers LEX ou DIAL.

L'indication de ces relations mères-filles permet de retrouver les différentes étapes du développement d'une théorie en observant les modifications qui sont réalisées lorsqu'on passe de la théorie-mère à la théorie-fille.

- A chaque théorie SSA est associé un type. En l'occurence, le type théorie SSA. (THSSA)

- A chaque théorie SSA est associé un score définissant son degré de vraisemblance. Ce score permet d'évaluer s'il vaut la peine de continuer à développer cette théorie.

Ces deux derniers attributs sont utilisés spécialement lors du développement de cette théorie.

6.8.3.3 Les théories Purement Sémantiques Descendantes (PSD).

- A chaque théorie PSD est associé un numéro identifiant.

- A chaque théorie PSD est associée une structure d'arbre sémantique .

- En général, chaque théorie PSD est fille d'une ou de plusieurs théories mères :

  - à l'exception des théories PSD initiales obtenues à partir des hypothèses émises par la composante DIAL; mais à part celles-ci :

  - chaque théorie PSD est fille, soit d'une théorie PSD dans le cas habituel, soit d'une théorie PSD mère et d'une théorie sémantique ascendante mère dans le cas d'un rattachement d'une théorie ascendante à une théorie descendante, soit d'une théorie SSD dans le cas d'un abandon de la partie syntaxique de la théorie.

- En règle générale, chaque théorie PSD est mère d'une ou de plusieurs théories PSD filles ou d'une théorie sémantique fille , à l'exception :

  - des théories englobant tout l'énoncé,

  - des théories abandonnées pour cause de score trop faible,

  - des théories en cours d'étude.

  Chaque théorie PSD fille a été obtenue en opérant une ou plusieurs modifications à la (aux) théorie(s) mère(s) :

  - soit parce qu'un ou plusieurs ajouts ont été effectués sur l'arbre sémantique de la théorie SSD mère. Ces modifications permettent au module SYN-SEM d'émettre au moins une hypothèse vers LEX ou DIAL.

  - soit en rattachant la théorie PSA mère à la théorie PSD mère

- soit en transformant la théorie SSD mère en théorie PSD par suppression de la partie syntaxique.

L'indication de ces relations mères-filles permet de retrouver les différentes étapes du développement d'une théorie en observant les modifications qui sont réalisées lorsqu'on passe de la théorie-mère à la théorie-fille.

- A chaque théorie PSD est associé un type. En l'occurence, le type théorie PSD. (THSD)

- A chaque théorie PSD est associé un score définissant son degré de vraisemblance. Ce score permet d'évaluer s'il vaut la peine de continuer à développer cette théorie.

Ces deux derniers attributs sont utilisés spécialement lors du développement de cette théorie.

6.8.3.4 Les théories Purement Sémantiques Ascendantes (PSA).

- A chaque théorie PSA est associé un numéro identifiant.

- A chaque théorie PSA est associée une structure d'arbre sémantique.

- En général, chaque théorie PSA est fille d'une ou de plusieurs théories syntaxico-sémantiques (ascendantes ou descendantes) mères :

  - à l'exception des théories PSA initiales obtenues à partir des hypothèses émises par la composante LEX; mais à part celles-ci :

  - chaque théorie PSA est fille, soit d'une théorie PSA dans le cas habituel, soit de deux théories PSA dans le cas d'un rattachement de la deuxième à la première, soit d'une théorie SSA dans le cas d'un abandon de la partie syntaxique de la théorie SSA.

- En règle générale, chaque théorie PSA est mère d'une ou de plusieurs théories PSA ou PSD filles, à l'exception :

  - des théories englobant tout l'énoncé,

  - des théories abandonnées pour cause de score trop faible,

  - des théories en cours d'étude.

  Chaque théorie PSA fille a été obtenue en opérant une ou plusieurs modifications dans la (les) théorie(s) mère(s) :

  - soit parce qu'un ou plusieurs ajouts ont été effectués à l'arbre sémantique de la théorie SSD mère. Ces modifications permettent au module SYN-SEM d'émettre au moins une hypothèse vers LEX ou DIAL.

  - soit en rattachant une théorie PSA à une autre théorie PSA.

  - soit en transformant la théorie SSA mère en théorie PSA par suppression de la partie syntaxique.

L'indication de ces relations mères-filles permet de retrouver les différentes étapes du développement d'une théorie en observant les modifications qui sont réalisées lorsqu'on passe de la théorie-mère à la théorie-fille.

- A chaque théorie PSA est associé un type. En l'occurence, le type théorie PSA. (THSA)

- A chaque théorie PSA est associé un score définissant son degré de vraisemblance. Ce score permet d'évaluer s'il vaut la peine de continuer à développer cette théorie.

Ces deux derniers attributs sont utilisés spécialement lors du développement de cette théorie.

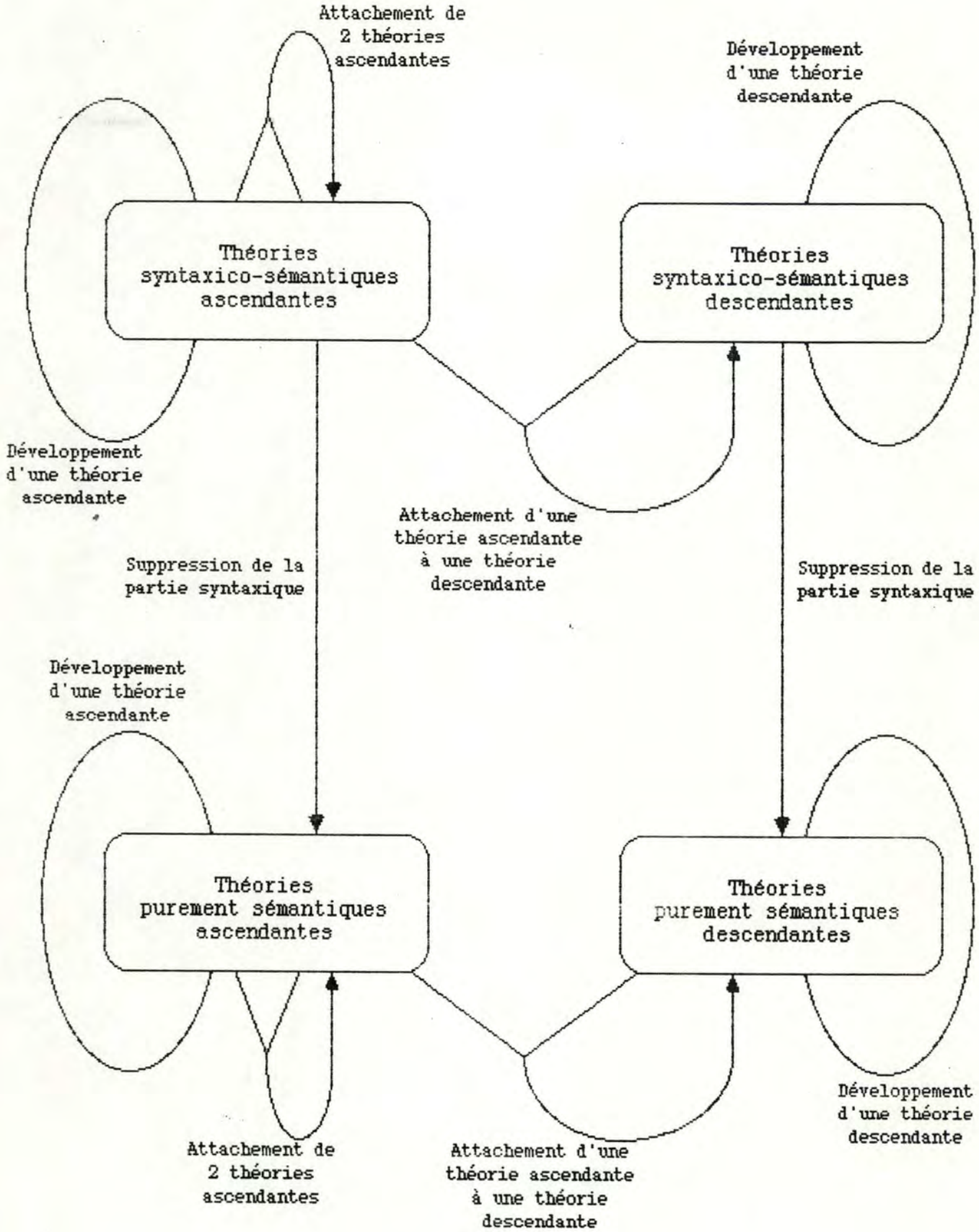
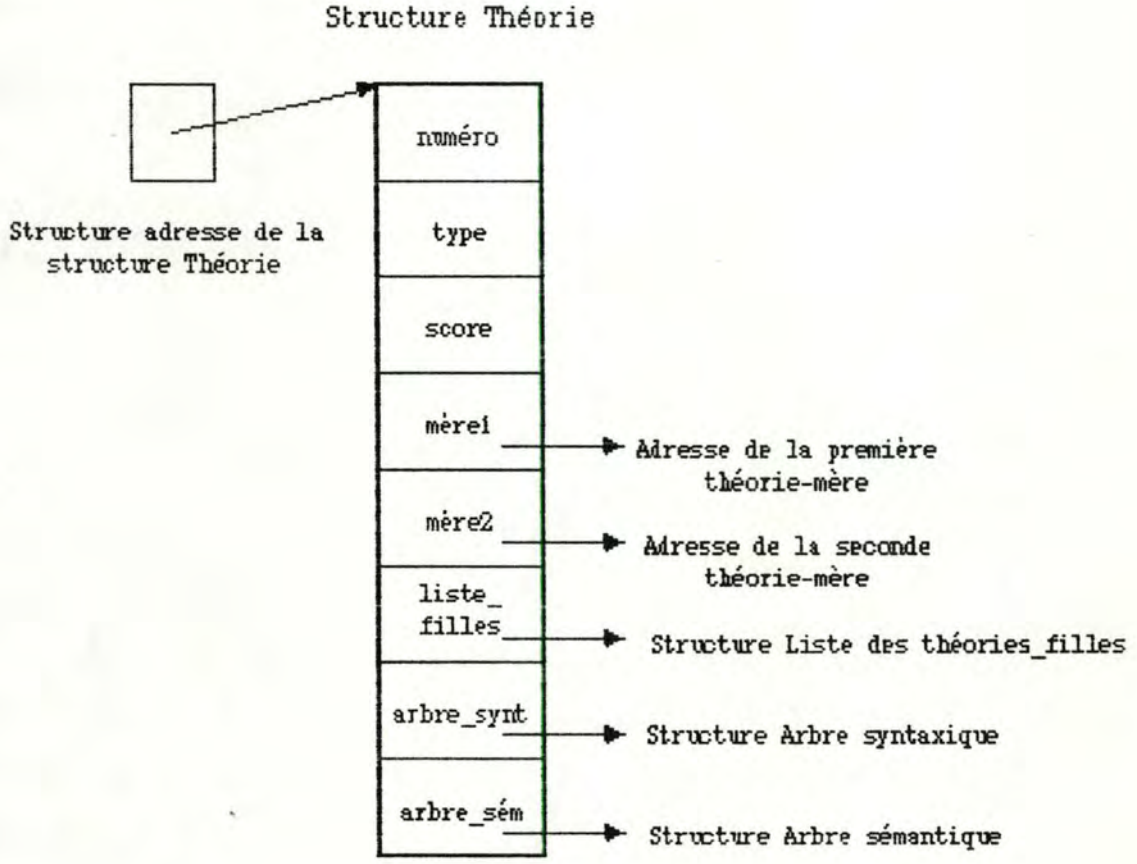


Figure VI.3 : Relations entre les théories-mères et les théories-filles

Une représentation graphique de la structure de données qui va être utilisé pour implémenter les théories est :



Après avoir expliqué ce qu'est une théorie et présenté ses différents attributs, je vais parler des autres structures de données qui sont utilisées par les analyseurs et stockées dans la base de théories.

6.8.3.5 Les arbres syntaxiques.

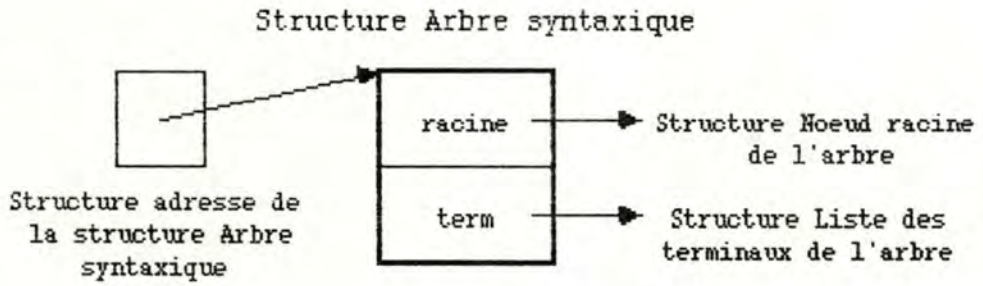
La structure de ces arbres ne pose pas de problème particulier. Les R.N.P. sont bien connus et la manière de construire ces arbres est maintenant bien définie.

La structure de données que nous allons utiliser pour représenter un arbre syntaxique ne contient que deux champs, à savoir :

• un *pointeur vers le noeud racine* de l'arbre syntaxique de la théorie (les autres noeuds de l'arbre peuvent être retrouvés en passant par ce noeud).

• un *pointeur vers une liste des terminaux reconnus* sur l'arbre syntaxique. Cette liste est employée par l'analyseur qui a besoin de retrouver rapidement les terminaux déjà reconnus, et cela sans devoir reparcourir l'arbre chaque fois.

La représentation graphique de cette structure est :



6.8.3.5.1 Les noeuds syntaxiques.

Un arbre est composé de noeuds. Il est donc nécessaire de représenter ces noeuds. Un noeud possède différents attributs permettant de l'identifier et le développement de la théorie à laquelle il appartient. Ces attributs sont :

• *un type* : un noeud peut être de type soit Indéterminé (NDSIND) qui est sa valeur par défaut, par exemple lorsqu'il vient d'être créé, soit Non-terminal de la grammaire (NDSNTERM), soit Terminal de la grammaire (NDSTERM), soit Réalisation d'un terminal (NDSREAL). En effet, il existe une différence entre la notion de terminal de la grammaire (par exemple : nom) et sa réalisation (par exemple : poire).

• *un symbole* : à chaque noeud est associé une chaîne de caractères qui contient sa valeur (par exemple : GN, Verbe, passeport).

• *noeud-père* : à chaque noeud est associé le noeud-père de ce noeud dans l'arbre syntaxique.

• *liste de noeuds-fils* : à chaque noeud est associé la liste des noeuds-fils de ce noeud dans l'arbre syntaxique de la théorie qui contient ce noeud.

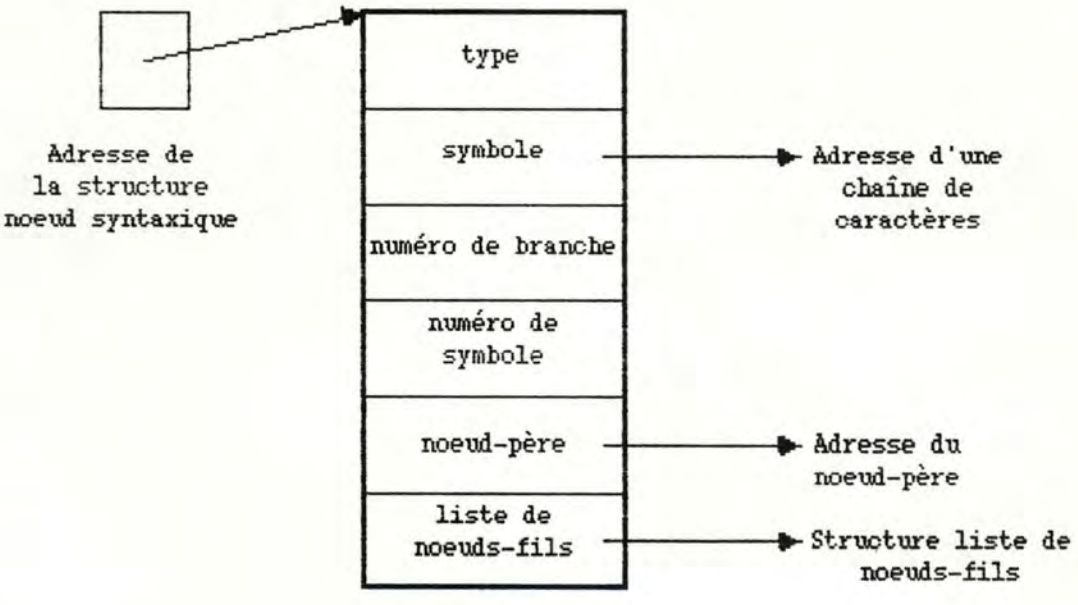
Les deux derniers attributs se rapportent aux R.N.P.. Ils permettent à l'analyseur de poursuivre le développement de l'arbre syntaxique. Ils indiquent le point de reprise de l'analyse dans les R.N.P..

• *numéro de branche* : il faut savoir lorsqu'on a un noeud, quelle branche du sous-réseau (sous-réseau qui est connu grâce au noeud-père) a permis de le produire => un entier numéro de branche (la numérotation existant dans la représentation interne des R.N.P.).

• *numéro de symbole* : il faut également connaître le numéro du symbole sur la branche (la numérotation existant dans la représentation interne des R.N.P.).

La représentation graphique d'un noeud syntaxique est :

Structure Noeud Syntaxique



6.8.3.6 Les arbres sémantiques.

Un arbre sémantique paraît fort semblable à un arbre syntaxique. Mais malheureusement, les notions utilisées dans la représentation sémantique et la manière de construire cette représentation sont fortement différentes. D'un point de vue des notions employées, la représentation sémantique est décrite en termes de proposition, prédicat, primitive, cas, ... plutôt que qu'en termes de terminal ou non-terminal de la grammaire. Les règles de construction de l'arbre sont de type condition-action et ces conditions sont évaluées en fonction du prédicat, de la primitive d'où provient ce cas, de la phrase nominale ainsi que de leurs environnements syntaxiques. L'analyseur sémantique utilise donc un autre type d'informations que celles utilisées par l'analyseur syntaxique pour représenter et développer un arbre sémantique. Pour ces raisons, il n'est pas possible d'utiliser une structure de données identique pour représenter un arbre sémantique et un arbre syntaxique.

Les structures de données qui seront présentées contiendront :

- des informations identifiant les éléments déjà découverts de la représentation sémantique .
- des informations nécessaires à l'analyseur pour poursuivre le développement des représentations sémantiques.

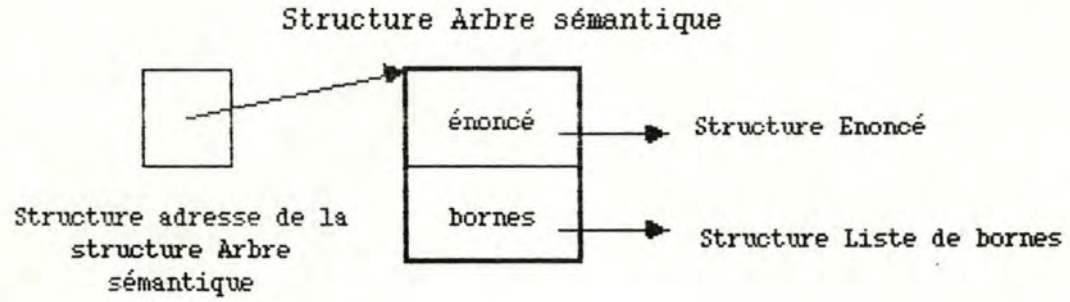
On remarque tout d'abord qu'une phrase est composée de propositions. On peut effectuer sur l'arbre sémantique un découpage similaire en propositions. L'arbre sémantique peut donc être considéré comme composé de noeuds correspondant chacun à une proposition. Une structure énoncé contiendra les diverses informations nécessaires à l'analyseur au sujet d'une proposition. La structure utilisée pour représenter un arbre sémantique est la suivante. A chaque structure d'arbre sémantique sont associés :

• La proposition principale de la phrase (une structure *énoncé*). Celle-ci correspond au noeud racine de l'arbre. On peut

retrouver tous les éléments de cet arbre en passant par ce noeud.

• La *liste des bornes des mots reconnus sur l'énoncé*. Cette liste est pratique car il ne faut pas tenter de reconnaître plusieurs mots à un même endroit de l'énoncé. Grâce à elle, on connaît directement les parties de l'énoncé déjà reconnues sans avoir à reparcourir tout l'arbre.

La représentation graphique de la structure d'arbre sémantique est :



6.8.3.6.1 Les énoncés.

L'arbre sémantique se décompose en différents noeuds correspondants aux multiples propositions de la phrase. Une structure énoncé contient

- des informations identifiant une proposition.
- des informations dépendant d'une proposition et nécessaires à l'analyseur pour la poursuite du développement des représentations.

A chaque structure énoncé sont associés divers attributs qui identifient une proposition :

• une *liste de modalités* qui est non-vide uniquement pour le noeud racine (proposition principale) de l'arbre.

• une *structure proposition* , laquelle est composée :

+ d'une *structure primitive* représentée par

- un champ *type* de la primitive : ce type est soit indéterminé (TPIND), soit un verbe (TPVERBE), soit un nom (TPNOM),

soit un adjectif (TPADJ).

- un champ *nom* : qui est le nom de la primitive, par exemple Agent, Objet, ...

- un champ *réalisation* : qui est sa valeur sous forme de chaîne de caractères, par exemple " Voiture, Pomme, Vouloir, ... "

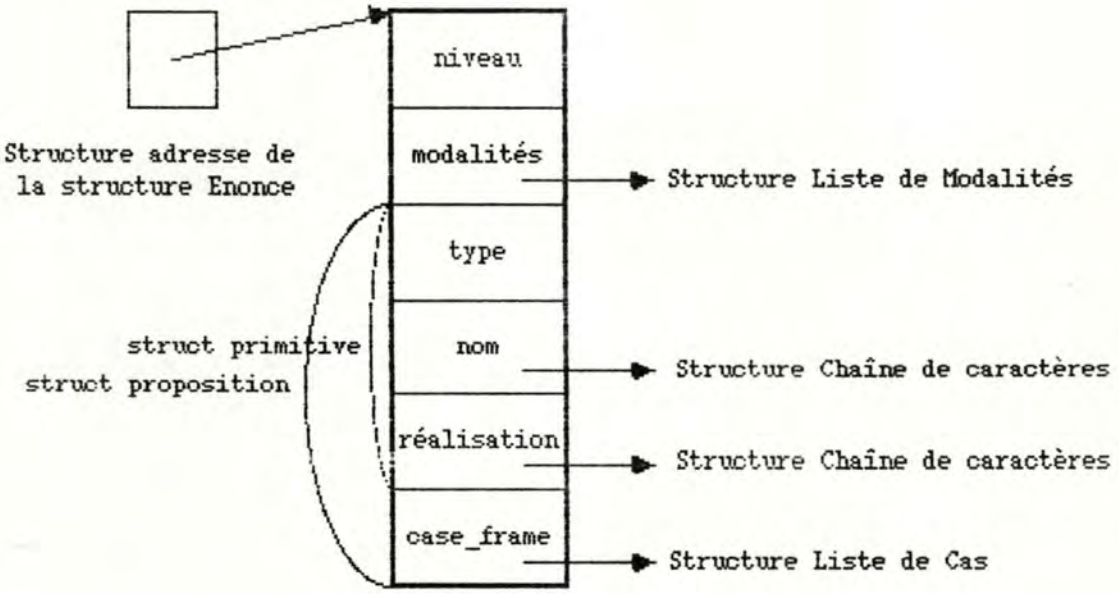
+ d'une *structure liste de cas* associée à cette primitive.

A chaque structure énoncé est associé un attribut supplémentaire, nécessaire à l'analyseur pour la poursuite du développement de la représentation sémantique :

- un *niveau* qui est défini comme :
  La proposition principale de la phrase possède le niveau 0.
  Le niveau d'une autre proposition = Le niveau de la proposition père + 1.

Représentation graphique de la structure Enoncé :

Structure Enoncé



6.8.3.6.2 Les structures de cas.

Les listes de cas associées aux primitives sont composées de

cas qui possèdent les attributs suivants :

• un *type* : qui est soit indéterminé (TCIND) qui est sa valeur par défaut lors de sa création, soit proposition nominale (TCPN), soit énoncé (TCENON).

• un *symbole* : qui est sa valeur sous forme de chaîne de caractères.

• une *réalisation* : cette réalisation peut se faire de 2 manières différentes,

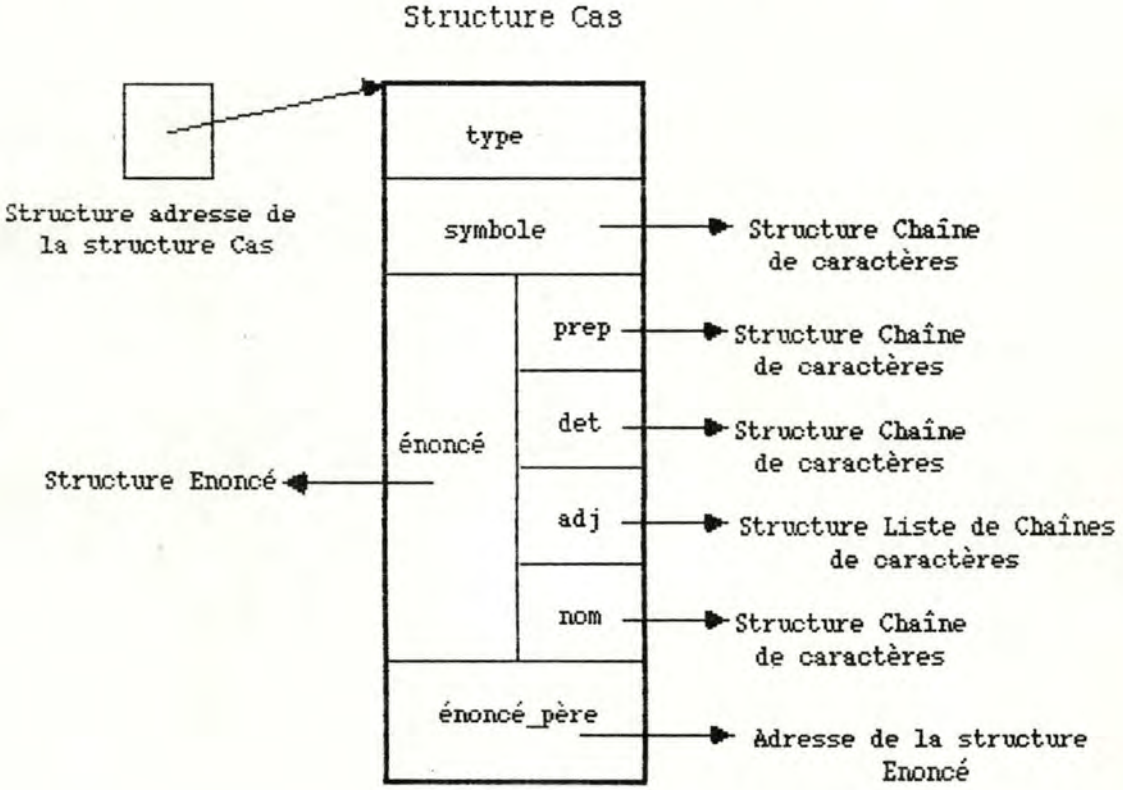
+ soit par tout un *énoncé de proposition* : on trouvera alors dans la partie réalisation une référence à une structure énoncé.

+ soit par un *groupe nominal* : on trouvera alors dans la partie réalisation une structure composée de

- une *préposition*.
- un *déterminant*.
- un *adjectif*.
- un *nom*.

• une référence au noeud *énoncé-père* duquel dépend ce cas et nécessaire au développement de la représentation sémantique.

Représentation graphique de la structure d'un cas :

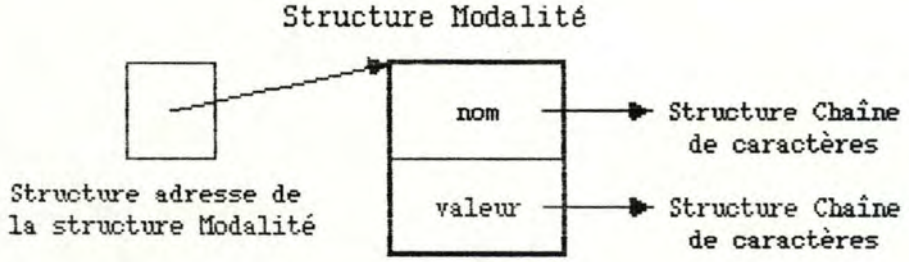


6.8.3.6.3 Les modalités.

Elles sont groupées dans une liste de modalités et chaque structure représentant une modalité comporte deux champs :

- son *nom* : par exemple, le temps, la voix, ...
- sa *valeur* : sous forme de chaîne de caractères.

Une représentation graphique d'une modalité est :

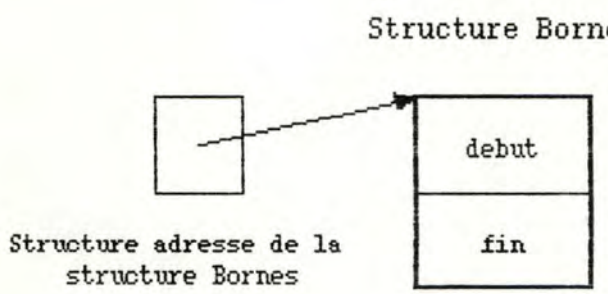


6.8.3.6.4 Les bornes.

La structure représentant une paire de bornes d'un mot est bien simple. Elle contient 2 champs qui sont :

- un *début* : qui marque le début de l'emplacement du mot sur l'énoncé de la phrase.
- une *fin* : qui marque la fin de l'emplacement du mot sur l'énoncé de la phrase.

Une représentation graphique de cette structure est :



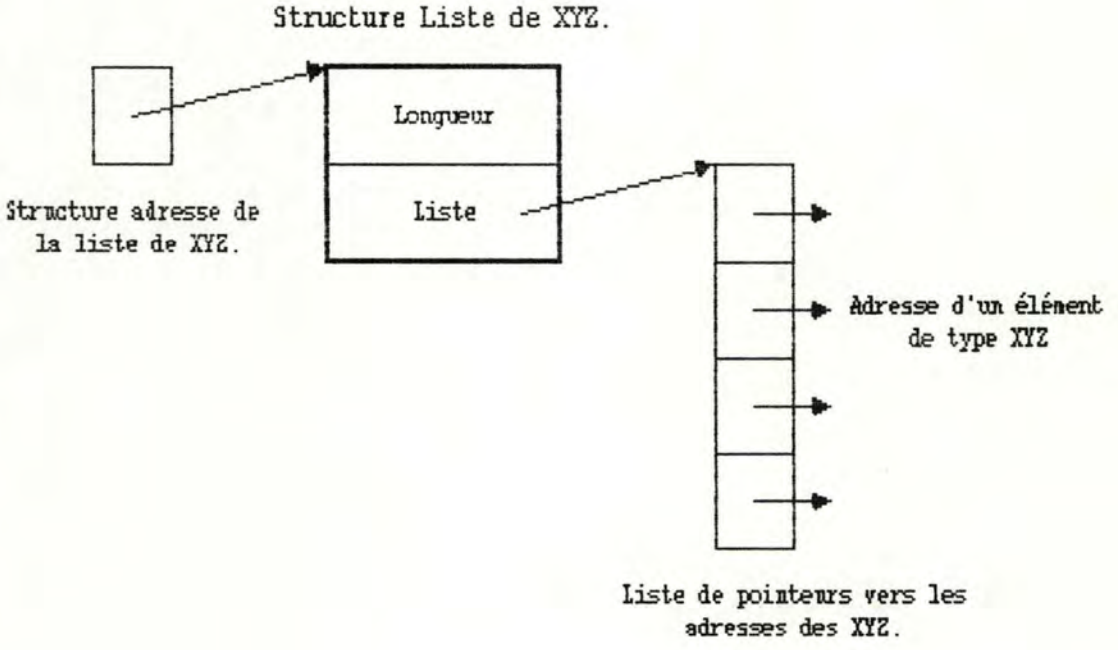
6.8.3.7 Les listes.

A plusieurs endroits, on parle de structure de liste. La structure utilisée pour représenter une liste est identique quel que soit le type des éléments de la liste. Prenons donc une liste d'éléments appelés pour la circonstance XYZ.

On a alors :

- à chaque liste est associée une *longueur* indiquant le nombre d'éléments de la liste.
- à chaque structure de liste est associée une référence à une liste d'adresses permettant de retrouver les éléments de la liste.

La représentation graphique d'une liste de XYZ est la suivante :



6.8.4 Remarque.

Il faut préciser maintenant que les structures qui ont été décrites ne sont peut-être pas celles qui seront utilisées dans le système final. En effet, la structure des données qui doivent être stockées par la base de théories, est identique à celui des données que traite le module principal de raisonnement du module SYN-SEM. La manière de construire les représentations syntaxiques des phrases est maintenant fixée bien qu'elle ait varié plusieurs fois au cours de lélaboration de notre travail. Les structures de données utilisées pour construire ces représentations sont donc bien définies. Par contre, la manière d'implémenter l'analyseur sémantique du module SYN-SEM n'est pas encore déterminée avec certitude. La dernière version des structures concernant les arbres sémantiques et qui vont vraisemblablement être utilisées date seulement de juillet 87. Il reste donc un certains nombre de détails concernant ces structures qui pourraient être changés dans un proche avenir. Ces changements de structure sont dus au fait que nous nous trouvons dans un domaine de recherche et qu'il faut régulièrement recommencer le travail pour trouver la solution qui paraît être la meilleure, la plus adéquate et qui permet de construire les représentations avec un minimum de problèmes.

6.9 Les fonctions d'accès à la base de théories.

Je viens de décrire les différentes structures qui sont utilisées par le module de raisonnement. Ces structures sont stockées dans la base de théories. Celle-ci constitue donc une base de données. Un certain nombre de fonctions d'accès sont nécessaires pour accéder à cette base. Ce sont principalement des fonctions de création, de destruction et de consultation des structures stockées dans la base. Ces fonctions d'accès dépendent donc directement du format des données à traiter.

La structure de ces données apparait maintenant comme quasiment définitive. Malheureusement, il n'en a pas toujours été ainsi et c'est pourquoi à chaque changement du format des données, il a

fallu recommencer la programmation des fonctions d'accès. Du point de vue des fonctions d'accès traitant la partie syntaxique des représentations, celles-ci ont été définies et réalisées, et on peut considérer celles-ci comme définitives. Néanmoins, il est vraisemblable qu'il faudra en créér l'une ou l'autre nouvelle mais celles qui seront véritablement utiles ne pourront être définies que lors de l'emploi réel de la base de théories.

Par contre, du point de vue des structures concernant les arbres sémantiques, celles qui viennent d'être présentées n'ont été adoptées que récemment (juillet 87) et diffèrent fortement des anciennes. Les fonctions qui avaient été définies puis réalisées ne sont donc plus d'actualité et le fait d'être limité par le temps pour la remise de ce mémoire ne m'a pas permis d'aller plus loin que la définition des nouvelles fonctions à réaliser.

Dans les pages qui suivent, je présenterai la définition des diverses fonctions d'accès qui ont été réalisées et la définition des structures en langage C qui ont été utilisées.

Dans les paragraphes suivants, on peut trouver la description de toutes les fonctions envisagées. On suppose que les notions d'ensemble, d'entier et de caractère sont connues du lecteur. Une liste sera pour nous un ensemble où les différents éléments ont un rang. C'est donc un ensemble où l'ordre d'énumération des éléments a une importance. La longueur d'une liste est son cardinal. De plus, nous dirons de deux éléments quelconques qu'ils sont identiques, non pas lorsqu'ils désignent le même objet physique mais, lorsque leurs composants sont identiques dans le cas d'objets structurés, ou lorsqu'ils ont la même valeur dans le cas d'objets élémentaires comme les caractères ou les entiers. Le signe '=' ne sera utilisé qu'entre deux désignations différentes d'un même objet physique. Pour chaque fonction d'accès, nous avons précisé son domaine de départ, son domaine d'arrivée, ses préconditions et postconditions si nécessaire ainsi que l'action qu'elle est censée effectuer. Les différents domaines seront notés de la façon suivante :

- CAR désigne l'ensemble des caractères.
- CHCAR désigne l'ensemble des chaînes de caractères.
  Une chaîne de caractères sera pour nous une liste de caractères
- LCHCAR désigne l'ensemble des listes de chaînes de caractères.

  $\forall$ lchcar $\in$ LCHCAR, lchcar $\in$ CHCAR.
- N désigne l'ensemble des entiers.
- NDS désigne l'ensemble des noeuds syntaxiques.
- TYPNDS désigne l'ensemble des types d'un noeud syntaxique.
- LNDS désigne l'ensemble des listes de noeuds syntaxiques.
- ARBSYN désigne l'ensemble des arbres syntaxiques.
- ARBSEM désigne l'ensemble des arbres sémantiques.
- THEORIE désigne l'ensemble des théories.
- TYPTH désigne le type d'une théorie.
- LTHEORIE désigne l'ensemble des théories.

  $\forall$ lthéorie $\in$ LTHEORIE, ltheorie $\subset$ THEORIE.

Fonctions définies sur tous les types d'objets.

Plusieurs fonctions sont définies sur tous les types d'objets,

ce sont les fonctions créer, détruire, copie et égal. Soit un ensemble X et un élément $x \in X$.

Constructeurs.

créer
Domaine de départ: /
Domaine d'arrivée: X
Préconditions: aucune.
Postconditions: $\exists!$ x $\in$ nouveau-X et $x \notin X$.
Actions: La fonction crée un x.

Simplificateurs.

détruire
Domaine de départ: X
Domaine d'arrivée: /
Préconditions: aucune
Postconditions: $\exists!$ x $\in$ ancien-X et $x \notin X$.
Actions: La fonction détruit le X.

copie
Domaine de départ: XxX.
Domaine d'arrivée: X.
Préconditions: aucune.
Postconditions: $\forall$ (x1,x2) $\in$ XxX,
copie(x1,x2) => identique(nouveau-x1,x2).
Actions: La fonction copie x1 dans x2.

Sélecteurs.

Domaine de départ: XxX.
Domaine d'arrivée: {identiques,différents}
Actions: La fonction indique si deux éléments de X sont identiques.

Fonctions définies sur les listes.

Certaines fonctions sont définies sur tous les objets de type liste. Ce sont les fonctions ajcop, ajout, inscop, inser, cat, long, vide, elen, elendup, loccel, cont. Soit l'ensemble LX des listes lx d'éléments $x \in X$.

Constructeurs.

ajout
Domaine de départ: LXxX.
Domaine d'arrivée: LX.
Préconditions: aucune.
Postconditions: $\forall$ (lx,x) $\in$ LXxX, ajout(lx,x) =>

∃! élément, élément ∈ nouveau-lx
et élément ∉ ancien-lx
et élément = x.

Actions: La fonction ajoute x à lx.

Simplificateurs

ajcop

Domaine de départ: LXxX.
Domaine d'arrivée: LX.
Préconditions: aucune.
Postconditions: ∀ (lx,x) ∈ LXxX, ajcop(lx,x) =>
∃! élément, élément ∈ nouveau-lx et
élément ∉ ancien-lx et
identique(élément,x) et
identique(rang(nouveau-lx,élément),longueur(lx)).
Actions: La fonction ajoute une copie de x à lx.

inscop

Domaine de départ: LXxXxN.
Domaine d'arrivée: LX.
Préconditions: aucune.
Postconditions: ∀ (lx,x) ∈ LXxX, inscop(lx,x) =>
∃! élément, élément ∈ nouveau-lx et
élément ∉ ancien-lx et
identique(élément,x) et
identique(rang(nouveau-lx,élément),min(longueur(lx),n)).
Actions: La fonction insère une copie de x à l'endroit n dans lx.

inser

Domaine de départ: LXxXxN.
Domaine d'arrivée: LX.
Préconditions: aucune.
Postconditions: ∀ (lx,x) ∈ LXxX, inser(lx,x) =>
∃! élément, élément ∈ nouveau-lx e élément ∉ ancien-lx et
élément = x et rang(lx,x) = min(longueur(nouveau-lx),n).

Actions: La fonction insère x à l'endroit n dans lx.

cat

Domaine de départ: LXxLX.
Domaine d'arrivée: LX.
Préconditions: aucune.
Postconditions: ∀ (lx1,lx2) ∈ LXxLX, cat(lx1,lx2) =>
∀ x ∈ ancien-lx1, x ∈ nouveau-lx1 et
rang(nouveau-lx1,x) = rang(ancien-lx1,x) et
∀ x ∈ lx2, x ∈ nouveau-lx1 et
rang(nouveau-lx1,x) = rang(lx2,x)+longueur(ancien-lx1)
Actions: La fonction concatène lx2 à lx1.

Sélecteurs.

long

Domaine de départ: LX.
Domaine d'arrivée: N.
Actions: La fonction indique la longueur de lx ∈ LX.

vide
Domaine de départ: LX.
Domaine d'arrivée: {vide,non-vide}.
Actions: La fonction indique si lx ∈ LX est vide.

elen
Domaine de départ: LXxN
Domaine d'arrivée: X
Actions: La fonction fournit l'élément de rang n de lx ∈ LX.

elendup
Domaine de départ: LXxN.
Domaine d'arrivée: X.
Actions: La fonction fournit une copie de l'élément de rang n de lx ∈ LX.

loccel
Domaine de départ: LXxX.
Domaine d'arrivée: LENT.
Actions: La fonction fournit la liste des rangs des éléments d'un lx ∈ LX identiques à un certain x ∈ X.

cont
Domaine de départ: LXxX.
Domaine d'arrivée: {contient,ne-contient-pas}
Actions: La fonction indique si lx ∈ LX contient un élément identique à x ∈ X.

Les autres fonctions qui ont été définies sont toutes particulières aux différents types d'objets.

Fonctions sur les noeuds syntaxiques.

Constructeurs.

deftyp_nds
Domaine de départ: NDSxTYPNDS.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,typ) ∈ NDSxTYPNDS,
deftyp_nds(nds,typ) =>
identique(type(nouveau-nds),typ).
Actions: La fonction définit le type du noeud par typ.

defsymb_nds
Domaine de départ: NDSxCHCAR.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,chcar) ∈ NDSxCHCAR,

defsymb_nds(nds,chcar) =>
identique(symbole(nouveau-nds),chcar).

Actions: La fonction définit le symbole du noeud par chcar.

defnb_nds

Domaine de départ: NDSxENT.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,num) ∈ NDSxENT,
defnb_nds(nds,num) =>
identique(numéro-de-branche(nouveau-nds),num).
Actions: La fonction définit le numéro de branche du noeud par num.

defns_nds

Domaine de départ: NDSxENT.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,num) ∈ NDSxENT,
defns_nds(nds,num) =>
identique(numéro-de-symbole(nouveau-nds),num).
Actions: La fonction définit le numéro de symbole du noeud par num.

defdeb_nds

Domaine de départ: NDSxfloat.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,num) ∈ NDSxfloat,
defnb_nds(nds,num) =>
identique(début(nouveau-nds),num).
Actions: La fonction définit le début du noeud par num.

deffin_nds

Domaine de départ: NDSxfloat.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds,num) ∈ NDSxfloat,
deffin_nds(nds,num) =>
identique(fin(nouveau-nds),num).
Actions: La fonction définit la fin du noeud par num.

defpere_nds

Domaine de départ: NDSxNDS.
Domaine d'arrivée: NDS.
Préconditions: aucune.
Postconditions: ∀ (nds1,nds2) ∈ NDSxNDS,
defpere_nds(nds1,nds2) =>
identique(nds-pere(nouveau-nds2),nds1).
Actions: La fonction définit le noeud-père du noeud nds2 par le noeud nds1.

deflf_nds

Domaine de départ: NDSxLNDS.
Domaine d'arrivée: NDS.
Préconditions: aucune.

Postconditions: ∀ (nds,lnds) ∈ NDSxLNDS,
deflf_nds(nds,lnds) =>
identique(nds-liste-fils(nouveau-nds),lnds).
Actions: La fonction définit la liste de fils du noeud nds par la liste lnds.

Sélecteurs.

type_nds
Domaine de départ: NDS.
Domaine d'arrivée: TYPNDS.
Actions: La fonction fournit le type typnds ∈ TYPNDS d'un noeud nds ∈ NDS.

symb_nds
Domaine de départ: NDS.
Domaine d'arrivée: CHCAR.
Actions: La fonction fournit le symbole chcar ∈ CHCAR d'un noeud nds ∈ NDS.

numbr_nds
Domaine de départ: NDS.
Domaine d'arrivée: ENT.
Actions: La fonction fournit le numéro de branche num ∈ ENT d'un noeud nds ∈ NDS.

numsym_nds
Domaine de départ: NDS.
Domaine d'arrivée: ENT.
Actions: La fonction fournit le numéro de symbole num ∈ ENT d'un noeud nds ∈ NDS.

deb_nds
Domaine de départ: NDS.
Domaine d'arrivée: float.
Actions: La fonction fournit le début deb ∈ float d'un noeud nds ∈ NDS.

fin_nds
Domaine de départ: NDS.
Domaine d'arrivée: float.
Actions: La fonction fournit la fin fn ∈ float d'un noeud nds ∈ NDS.

ndpere_nds
Domaine de départ: NDS.
Domaine d'arrivée: NDS.
Actions: La fonction fournit le noeud-père ndpère ∈ NDS d'un noeud nds ∈ NDS.

listef_nds
Domaine de départ: NDS.

Domaine d'arrivée: LNDS.

Actions: La fonction fournit la liste des noeuds-fils lnds ∈ LNDS d'un noeud nds ∈ NDS.

Fonctions définies sur les arbres syntaxiques.

Constructeurs.

defrac_arbsyn
Domaine de départ: ARBSYNxNDS.
Domaine d'arrivée: ARBSYN.
Préconditions: aucune.
Postconditions: ∀ (arbsyn,nds) ∈ ARBSYNxNDS,
defrac_arbsyn(arbsyn,nds) =>
identique(racine-arbsyn(nouveau-arbsyn),nds).
Actions: La fonction définit le noeud-racine de l'arbre syntaxique par le noeud nds.

deft_arbsyn
Domaine de départ: ARBSYNxLNDS.
Domaine d'arrivée: ARBSYN.
Préconditions: aucune.
Postconditions: ∀ (arbsyn,lnds) ∈ ARBSYNxLNDS,
deft_arbsyn(arbsyn,lnds) =>
identique(liste-des terminaux-arbsyn(nouveau-arbsyn),nds).
Actions: La fonction définit la liste des terminaux de l'arbre syntaxique par la liste lnds.

Sélecteurs.

rac_arbsyn
Domaine de départ: ARBSYN.
Domaine d'arrivée: NDS.
Actions: La fonction fournit le noeud racine nds ∈ NDS d'un arbre syntaxique arbsyn ∈ ARBSYN.

term_arbsyn
Domaine de départ: ARBSYN.
Domaine d'arrivée: LNDS.
Actions: La fonction fournit la liste des terminaux lnds ∈ LNDS d'un arbre syntaxique arbsyn ∈ ARBSYN.

Fonctions définies sur les théories.

Constructeurs.

defnum_théorie
Domaine de départ: THEORIExENT.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie,num) ∈ THEORIExENT,

defnum_théorie(théorie,num) =>
identique(numéro-de-théorie(nouveau-théorie),num).
Actions: La fonction définit le numéro de la théorie par num.

deftyp_théorie
Domaine de départ: THEORIExTYPTH.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie,typ) ∈ THEORIExTYPTH,
deftyp_théorie(théorie,typ) =>
identique(type-de-théorie(nouveau-théorie),typ).
Actions: La fonction définit le type de la théorie par typ.

defsc_théorie
Domaine de départ: THEORIExdouble.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie,score) ∈ THEORIExdouble,
defsc_théorie(théorie,score) =>
identique(score-de-la-théorie(nouveau-théorie),score).
Actions: La fonction définit le score de la théorie par score.

defmr1_théorie
Domaine de départ: THEORIExTHEORIE.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie1,theorie2) ∈ THEORIExTHEORIE,
defmr1_théorie(théorie1,theorie2) =>
identique(theorie-mere1-de-théorie(nouveau-théorie1),théorie2).
Actions: La fonction définit la première théorie-mère de la théorie1 ∈ THEORIE par théorie2 ∈ THEORIE.

defmr2_théorie
Domaine de départ: THEORIExTHEORIE.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie1,theorie2) ∈ THEORIExTHEORIE,
defmr2_théorie(théorie1,theorie2) =>
identique(theorie-mere2-de-théorie(nouveau-théorie1),théorie2).
Actions: La fonction définit la deuxième théorie-mère de la théorie1 ∈ THEORIE par la théorie2 ∈ THEORIE.

deflf_théorie
Domaine de départ: THEORIExLTHEORIE.
Domaine d'arrivée: THEORIE.
Préconditions: aucune.
Postconditions: ∀ (théorie,ltheorie) ∈ THEORIExLTHEORIE,
deflf_théorie(théorie,ltheorie) =>
identique(liste-de-théorie-filles-de-théorie(nouveau-théorie1),lthéorie).
Actions: La fonction définit la liste de théories-filles de la théorie théorie ∈ THEORIE par lthéorie ∈ LTHEORIE.

defasyn_théorie
Domaine de départ: THEORIExARBSYN.
Domaine d'arrivée: THEORIE.

Préconditions: aucune.

Postconditions: ∀ (théorie,arbsyn) ∈ THEORIExARBSYN,
defasyn_théorie(théorie,asyn) =>
identique(arbre-syntaxique-de-théorie(nouveau-théorie),arbsyn).

Actions: La fonction définit l'arbre syntaxique de la théorie par arbsyn.

defasem_théorie

Domaine de départ: THEORIExARBSEM.

Domaine d'arrivée: THEORIE.

Préconditions: aucune.

Postconditions: ∀ (théorie,arbsem) ∈ THEORIExARBSEM,
defasem_théorie(théorie,asem) =>
identique(arbre-sémantique-de-théorie(nouveau-théorie),arbsem).

Actions: La fonction définit l'arbre sémantique de la théorie par arbsem.

Sélecteurs.

num_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: ENT.

Actions: La fonction fournit le numéro de la théorie num ∈ ENT d'une théorie théorie ∈ THEORIE.

type_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: TYPTH.

Actions: La fonction fournit le type de la théorie typ ∈ TYPTH d'une théorie théorie ∈ THEORIE.

score_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: double.

Actions: La fonction fournit le score de la théorie score ∈ double d'une théorie théorie ∈ THEORIE.

mère1_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: THEORIE.

Actions: La fonction fournit la première théorie-mère de la théorie théorie ∈ THEORIE d'une théorie théorie ∈ THEORIE.

mère2_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: THEORIE.

Actions: La fonction fournit la deuxième théorie-mère de la théorie théorie ∈ THEORIE d'une théorie théorie ∈ THEORIE.

arbsyn_théorie

Domaine de départ: THEORIE.

Domaine d'arrivée: ARBSYN.

Actions: La fonction fournit l'arbre syntaxique arbsyn ∈ ARBSYN de la théorie théorie ∈ THEORIE.

arbsem_théorie
Domaine de départ: THEORIE.
Domaine d'arrivée: ARBSEM.
Actions: La fonction fournit l'arbre sémantique arbsem ∈ ARBSEM de la théorie théorie ∈ THEORIE.

listef_théorie
Domaine de départ: THEORIE.
Domaine d'arrivée: LTHEORIE.
Actions: La fonction fournit la liste de théories-filles lthéories ∈ LTHEORIE de la théorie théorie ∈ THEORIE.

6.10 Implémentation des structures et des fonctions d'accès.

Pour l'implémentation de notre système, nous avons choisi le langage C, les structures et les fonctions d'accès sont donc également implémentées en C. Les structures seront données de façon détaillée mais en ce qui concerne les fonctions, nous nous sommes limités à fournir les entêtes des fonctions avec la description des paramètres d'appel.

6.10.1 Structures.

La définition en langage C des structures est la suivante :

Chaîne de caractères:

```
typedef struct Chcar
     {
       int longueur;
       char *chaîne;
     }          *CHCAR;
```

Liste de chaînes de caractères:

```
typedef struct Lchcar
     {
       int longueur;
       CHCAR **liste;
     }          *LCHCAR;
```

Liste d'entiers:

```
typedef struct Lent
     {
       int longueur;
       int *liste;
     }          *LENT;
```

Type de noeud syntaxique:

```
typedef enum {NDSIND, NDSREAL, NDSTERM, NDSNTERM} TYPNDS;
```

Noeud syntaxique:

```
typedef struct Nds
     {
       TYPNDS type;
       CHCAR symbole;
       int num_br;
       int num_symb;
       float debut;
       float fin;
       struct Nds **ndspere;
       struct Lnds *liste_fils;
     }          *NDS;
```

Liste de noeuds syntaxiques:

```
typedef struct Lnds
     {
       int longueur;
       NDS **liste;
     }          *LNDS;
```

Arbre syntaxique:

```
typedef struct Arbsyn
     {
       NDS racine;
       LNDS term;
     }          *ARBSYN;
```

Modalité:

```
typedef struct Mod
     {
       CHCAR nom;
       CHCAR valeur;
     }            *MOD;
```

Liste de modalités:

```
typedef struct Lmod
     {
       int longueur;
       MOD **liste;
     }            *LMOD;
```

Type de cas:

```
typedef enum Typcase { TCIND, TCPN, TCENON } TYPCASE;
```

Cas:

```
typedef struct Case
     {
       TYPCASE type;
       CHCAR symbole;
       struct enonce **enonce_pere;
       union realisation;
          {
            struct enonce;
            struct gnom;
               {
                 CHCAR prep;
                 LCHCAR det;
                 LCHCAR adj;
                 CHCAR nom;
               }
          }
     }          *CASE;
```

Liste de cas:

```
typedef struct Lcase
     {
       int longueur;
       CASE **liste;
     }          *LCASE;
```

Le type type de primitive:

```
typedef enum Typprim { TPIND, TPVERBE, TPNOM, TPADJ } TYPPRIM;
```

Enonce:

```
typedef struct Enoncé

     {
       int niveau;
       LMOD modalités;
       struct Cas **cas_réalisé;
       struct proposition
```

```
        {
          struct primitive
            {
              TYPPRIM type;
              CHCAR nom;
              CHCAR real;
            }
          LCASE caseframe;
        }
    }       *ENONCE;
```

Bornes:

```
typedef struct Borne
    {
      float debut;
      float fin;
    }     *BORNE;
```

Liste de bornes:

```
typedef struct Lborne
    {
      int longueur;
      BORNE **liste;
    }     *LBORNE;
```

Arbre sémantique:

```
typedef struct Arbsem

    {
      ENONCE enonce;
      LBORNES bornes;
    }
```

Type de théories:

```
typedef enum Typth {THIND, THSSD, THSSA, THSD, THSA } TYPTH;
```

Théorie:

```
typedef struct Théorie

    {
      int numéro;
      TYPTH type;
      double score;
      struct Théorie **mère1;
      struct Théorie **mère2;
      struct Lthéorie *liste_filles;
      ARBSYN arbre_synt;
      ARBSEM arbre_sem;
    }     *THEORIE;
```

Liste de théories:

```
typedef struct Lthéorie
    {
      int longueur;
```

```
    THEORIE **liste;
}       *LTHEORIE;
```

6.10.2 Fonctions d'accès.

On trouvera dans ce paragraphe la liste des fonctions d'accès à la base de théories concernant les structures syntaxiques et les théories. Pour chaque fonction, nous avons fourni son nom, ses paramètres d'appel ainsi qu'une courte description de ses effets.

De plus, afin de faciliter certaines manipulations, nous avons ajouté quelques fonctions qui ne sont pas définies dans la partie précedente, s'agissant de problèmes trop proches du niveau implémentation. Ce sont essentiellement des fonctions d'entrée-sortie qui ont été définies pour toutes les fonctions.

Definitions des fonctions d'acces realisees.
==========

1. Fonctions d'acces sur les noeuds syntaxiques.

+ NDS creer_nds(&nds) NDS nds;
Creation d'un noeud vide.

+ NDS detr_nds(&nds) NDS nds;
Destruction du noeud nds et de sa descendance.

+ NDS copie_nds(&nds1,nds2) NDS nds1,nds2;
Copie du noeud nds2 et de sa descendance dans le noeud nds1.

+ NDS deftyp_nds(&nds,type) NDS nds;
TYPNDS type;
Def. du type du noeud nds.

+ NDS defsymb_nds(&nds,symb) NDS nds;
CHCAR symb;
Def. du symbole du noeud nds.

+ NDS defnb_nds(&nds,numero) NDS nds;
int numero;
Def. du numero de la branche du noeud nds.

+ NDS defns_nds(&nds,numero) NDS nds;
int numero;
Def. du numero du symbole du noeud nds.

+ NDS defdeb_nds(&nds,debut) NDS nds;
float debut;
Def. du debut du noeud nds sur l'enonce.

+ NDS deffin_nds(&nds,fin) NDS nds;
float fin;
Def. de la fin du noeud nds sur l'enonce.

+ NDS defpere_nds(&nds1,&nds2) NDS nds1,nds2;
Def. du noeud-pere nds2 du noeud nds1.

+ NDS deflf_nds(&nds,liste) NDS nds;
LNDS liste;
Def. de la liste des noeuds-fils du noeud nds.

+ ecr_nds(nds) NDS nds;
Ecriture du noeud nds sur le fichier standard en sortie.

+ fecr_nds(fichier,nds) FILE *fichier;
NDS nds;
Ecriture du noeud nds sur le fichier fichier en sortie.

+ NDS lire_nds(&nds) NDS nds;
Lecture du noeud nds sur le fichier standard en entree.

+ NDS flire_nds(fichier,&nds) FILE *fichier;
NDS nds;
Lecture du noeud nds sur le fichier fichier en entree.

+ TYPNDS type_nds(nds) NDS nds;
Type du noeud nds.

+ CHCAR symb_nds(nds) NDS nds;
Symbole du noeud nds.

+ int numbr_nds(nds) NDS nds;
Numero de la branche du noeud nds.

+ int numsym_nds(nds) NDS nds;
Numero du symbole du noeud nds.

+ float deb_nds(nds) NDS nds;
Debut du noeud nds sur l'enonce.

+ float fin_nds(nds) NDS nds;
Fin du noeud nds sur l'enonce.

+ NDS ndpere_nds(nds) NDS nds;
Noeud-pere du noeud nds.

+ LNDS listef_nds(nds) NDS nds;
Liste des noeuds-fils du noeud nds.

+ BOOLEAN egal_nds(nds1,nds2) NDS nds1,nds2;
Comparaison de 2 noeuds nds1 et nds2 et de leur descendance.

2. Fonctions d'acces sur les listes de noeuds syntaxiques.

+ LNDS creer_lnds(&lnds) LNDS lnds;
Creation d'une liste de noeuds vide.

+ LNDS detrl_lnds(&lnds) LNDS lnds;
Destruction de la structure liste de noeuds lnds.

+ LNDS detr_lnds(&lnds) LNDS lnds;
Destruction de la liste de noeuds lnds, des noeuds references par la liste et de la descendance de ces noeuds.

+ LNDS ajcop_lnds(&lnds,nds) LNDS lnds;
NDS nds;
Ajout de la duplication du noeud nds a la liste de noeuds lnds.

+ LNDS ajout_lnds(&lnds,&nds) LNDS lnds;
NDS nds;
Ajout du noeud nds a la liste de noeuds lnds.

+ LNDS inscop_lnds(&lnds,n,nds) LNDS lnds;
int n;
NDS nds;
Insertion de la duplication du noeud nds dans la liste de noeuds lnds a la nieme place.

+ LNDS inser_lnds(&lnds,n,&nds) LNDS lnds;
int n;
NDS nds;
Insertion du noeud nds dans la liste de noeuds lnds a la nieme place.

+ LNDS copie_lnds(&lnds1,lnds2) LNDS lnds1,lnds2;
Copie de la liste de noeuds lnds2 dans la liste de noeuds lnds1.

+ LNDS cat_lnds(&lnds1,lnds2) LNDS lnds1,lnds2;
Concatenation de la liste de noeuds lnds2 a la liste de noeuds lnds1.

+ LNDS supprnf_lnds(&lnds,n) LNDS lnds;

```
                                                        int n;
        Suppression de la reference a un noeud syntaxique se trouvant a
        l'endroit n dans la liste de noeuds syntaxiques lnds.

+ LNDS  supprel_lnds(&lnds,n)                           LNDS lnds;
                                                        int n;
        Suppression de la reference a un noeud syntaxique se trouvant a
        l'endroit n dans la liste de noeuds syntaxiques lnds et destruction
        de ce noeud reference, ainsi que de sa descendance.

+        ecr_lnds(lnds)                                 LNDS lnds;
        Ecriture de la liste de noeuds lnds sur le fichier standard en sortie.

+        fecr_lnds(fichier,lnds)                        FILE *fichier;
                                                        LNDS lnds;
        Ecriture de la liste de noeuds lnds sur le fichier fichier en sortie.

+ LNDS  lire_lnds(&lnds)                                LNDS lnds;
        Lecture de la liste de noeuds lnds sur le fichier standard en entree.

+ LNDS  flire(fichier,&lnds)                            FILE *fichier;
                                                        LNDS lnds;
        Lecture de la liste de noeuds lnds sur le fichier fichier en entree.

+ int     long_lnds(lnds)                               LNDS lnds;
        Longueur de la liste de noeuds lnds.

+ BOOLEAN  vide_lnds(lnds)                              LNDS lnds;
        Liste de noeuds lnds vide ?

+ NDS   elen_lnds(lnds,n)                               LNDS lnds;
                                                        int n;
        Nieme element de la liste de noeuds lnds.

+ NDS   elendup_lnds(lnds,n)                            LNDS lnds;
                                                        int n;
        Duplication du nieme element de la liste de noeuds lnds.

+ LENT   loccel_lnds(lnds,nds)                          LNDS lnds;
                                                        NDS nds;
        Liste des occurences du noeud nds dans la liste de noeuds lnds.

+ BOOLEAN  cont_lnds(lnds,nds)                          LNDS lnds;
                                                        NDS nds;
        Noeud nds contenu dans la liste de noeuds lnds ?

+ BOOLEAN  egal_lnds(lnds1,lnds2)                       LNDS lnds1,lnds2;
        Comparaison de deux listes de noeuds lnds1 et lnds2 ( = les noeuds +
        leur descendance ).
```

3. Fonctions d'acces sur les arbres syntaxiques.

```
+ ARBSYN  creer_arbsyn(&arbsyn)                         ARBSYN arbsyn;
        Creation d'un arbsyn vide.

+ ARBSYN  detr_arbsyn(&arbsyn)                          ARBSYN arbsyn;
        Destruction de l'arbsyn arbsyn.

+ ARBSYN  copie_arbsyn(&arbsyn1,arbsyn2)                ARBSYN arbsyn1,arbsyn2
        Copie de l'arbsyn arbsyn2 sur l'arbsyn arbsyn1.

+ ARBSYN  defrac_arbsyn(&arbsyn,racine)                 ARBSYN arbsyn;
                                                        NDS racine;
```

Def. de la racine de l'arbsyn arbsyn.

+ ARBSYN deft_arbsyn(&arbsyn,lterm) ARBSYN arbsyn;
LNDS lterm;
Def. de la liste de terminaux de l'arbsyn arbsyn.

+ ecr_arbsyn(arbsyn) ARBSYN arbsyn;
Ecriture de l'arbsyn arbsyn sur le fichier standard en sortie.

+ fecr_arbsyn(fichier,arbsyn) FILE *fichier;
ARBSYN arbsyn;
Ecriture de l'arbsyn arbsyn sur le fichier fichier en sortie.

+ ARBSYN lire_arbsyn(&arbsyn) ARBSYN arbsyn;
Lecture de l'arbsyn arbsyn sur le fichier standard en entree.

+ ARBSYN flire_arbsyn(fichier,&arbsyn) FILE *fichier;
ARBSYN arbsyn;
Lecture de l'arbsyn arbsyn sur le fichier fichier en entree.

+ NDS rac_arbsyn(arbsyn) ARBSYN arbsyn;
Racine de l'arbsyn arbsyn.

+ LNDS term_arbsyn(arbsyn) ARBSYN arbsyn;
Liste des terminaux de l'arbsyn arbsyn.

+ BOOLEAN egal_arbsyn(arbsyn1,arbsyn2) ARBSYN arbsyn1,arbsyn2
Comparaison de deux arbsyns arbsyn1 et arbsyn2.

4. Fonctions d'acces sur les theories.

+ THEORIE creer_theorie(&theorie) THEORIE theorie;
Creation d'une theorie vide.

+ THEORIE detr_theorie(&theorie) THEORIE theorie;
Destruction d'une theorie theorie et de sa descendance.

+ THEORIE copie_theorie(&theorie1,theorie2) THEORIE theorie1,
theorie2;
Copie de la theorie theorie2 et de sa descendance dans une
theorie theorie1.

+ THEORIE defnum_theorie(&theorie,num) THEORIE theorie;
int num;
Def. du numero de la theorie theorie.

+ THEORIE deftyp_theorie(&theorie,type) THEORIE theorie;
TYPTH type;
Def. du type de la theorie theorie.

+ THEORIE defsc_theorie(&theorie,score) THEORIE theorie;
double score;
Def. du score score de la theorie theorie.

+ THEORIE defmr1_theorie(&theorie1,&theorie2) THEORIE theorie1,
theorie2;
Def. de la premiere theorie-mere theorie2 de la theorie theorie1.

+ THEORIE defmr2_theorie(&theorie1,&theorie2) THEORIE theorie1,
theorie2;
Def. de la deuxieme theorie-mere theorie2 de la theorie theorie1.

+ THEORIE defilf_theorie(&theorie,ltheorie) THEORIE theorie;

```
                                                        LTHEORIE ltheorie;
      Def. de la liste de theories-filles ltheorie de la theorie theorie.

+ THEORIE  defasyn_theorie(&theorie,asyn)              THEORIE theorie;
                                                       ARBSYN asyn;
      Def. de l'arbsyn syntaxique de la theorie theorie.

+ THEORIE  defasem_theorie(&theorie,asem)              THEORIE theorie;
                                                       ARBSYN asem;
      Def. de l'arbsyn semantique de la theorie theorie.

+          ecr_theorie(theorie)                       THEORIE theorie;
      Ecriture de la theorie theorie sur le fichier standard en sortie.

+          fecr_theorie(fichier,theorie)              FILE *fichier;
                                                       THEORIE theorie;
      Ecriture de la theorie theorie sur le fichier fichier en sortie.

+ THEORIE  lire_theorie(&theorie)                      THEORIE theorie;
      Lecture de la theorie theorie sur le fichier standard en entree.

+ THEORIE  flire_theorie(fichier,&theorie)             FILE *fichier;
                                                       THEORIE theorie;
      Lecture de la theorie theorie sur le fichier fichier en entree.

+ int      num_theorie(theorie)                       THEORIE theorie;
      Numero de la theorie theorie.

+ TYPTH    type_theorie(theorie)                      THEORIE theorie;
      Type de la theorie theorie.

+ double score_theorie(theorie)                       THEORIE theorie;
      Score de la theorie theorie.

+ THEORIE  mere1_theorie(theorie)                     THEORIE theorie;
      Premiere theorie-mere de la theorie theorie.

+ THEORIE  mere2_theorie(theorie)                     THEORIE theorie;
      Deuxieme theorie-mere de la theorie theorie.

+ LTHEORIE listef_theorie(theorie)                    THEORIE theorie;
      Liste des theories-filles de la theorie theorie.

+ ARBSYN   asyn_theorie(theorie)                      THEORIE theorie;
      Arbsyn syntaxique associe a la theorie theorie.

+ ARBSYN   asem_theorie(theorie)                      THEORIE theorie;
      Arbsyn semantique associe a la theorie theorie.

+ BOOLEAN egal_theorie(theorie1,theorie2)             THEORIE theorie1,
                                                             theorie2;
      Comparaison de deux theories theorie1 et theorie2.
```

5. Fonctions d'acces sur les listes de theories.

```
+ LTHEORIE  creer_ltheorie(&ltheorie)                 LTHEORIE ltheorie;
      Creation d'une liste de theories vide.

+ LTHEORIE  detr_ltheorie(&ltheorie)                  LTHEORIE ltheorie;
      Destruction de la liste de theories ltheorie, des theories
      referencees et de leur descendance.
```

Destruction de la structure liste de theories ltheorie.

+ LTHEORIE ajcop_ltheorie(&ltheorie,theorie) LTHEORIE ltheorie;
THEORIE theorie;

Ajout de la duplication de la theorie theorie a la liste de theories ltheorie.

+ LTHEORIE ajout_ltheorie(&ltheorie,&theorie) LTHEORIE ltheorie;
THEORIE theorie;

Ajout de la theorie theorie a la liste de theories ltheorie.

+ LTHEORIE inscop_ltheorie(&ltheorie,n,theorie) LTHEORIE ltheorie;
int n;
THEORIE theorie;

Insertion de la duplication de la theorie theorie dans la liste de theories a l'endroit n.

+ LTHEORIE inser_ltheorie(&ltheorie,n,&theorie) THEORIE theorie;
int n;
LTHEORIE ltheorie;

Insertion d'une theorie theorie dans la liste de theories ltheorie a la nieme place.

+ LTHEORIE copie_ltheorie(&ltheorie1,ltheorie2) LTHEORIE ltheorie1,
ltheorie2;

Copie de la liste de theories ltheorie2 dans la liste de theories ltheorie1.

+ LTHEORIE cat_ltheorie(&ltheorie1,ltheorie2) LTHEORIE ltheorie1,
ltheorie2;

Concatenation de la liste de theories ltheorie2 a la liste de theories ltheorie1.

+ LTHEORIE suppref_ltheorie(&ltheorie,n) LTHEORIE ltheorie;
int n;

Suppression de la reference a une theorie se trouvant a l'endroit n dans la liste de theories ltheorie.

+ LTHEORIE suppre1_ltheorie(&ltheorie,n) LTHEORIE ltheorie;
int n;

Suppression de la reference a une theorie se trouvant a l'endroit n dans la liste de theories ltheorie et destruction de cette theorie referencee, ainsi que de sa descendance.

+ ecr_ltheorie(ltheorie) LTHEORIE ltheorie;

Ecriture de la liste de theories ltheorie sur le fichier standard en sortie.

+ fecr_ltheorie(fichier,ltheorie) FILE *fichier;
LTHEORIE ltheorie;

Ecriture de la liste de theories ltheorie sur le fichier fichier en sortie.

+ LTHEORIE lire_ltheorie(&ltheorie) LTHEORIE ltheorie;

Lecture de la liste de theories ltheorie sur le fichier standard en entree.

+ LTHEORIE flire(fichier,&ltheorie) FILE *fichier;
LTHEORIE ltheorie;

Lecture de la liste de theories ltheorie sur le fichier fichier en entree.

+ int long_ltheorie(ltheorie) LTHEORIE ltheorie;

Longueur de la liste de theories ltheorie.

+ BOOLEAN vide_ltheorie(ltheorie) LTHEORIE ltheorie;

Liste de theories ltheorie vide ?

+ THEORIE elen_ltheorie(ltheorie,n) LTHEORIE ltheorie;
int n;

Nieme element de la liste de theories ltheorie.

+ THEORIE elendup_ltheorie(ltheorie,n) LTHEORIE ltheorie;
int n;

Duplication du nieme element de la liste de theories ltheorie.

+ LENT loccel_ltheorie(ltheorie,theorie) LTHEORIE ltheorie;
THEORIE theorie;

Liste des occurences d'une theorie theorie dans la liste de theories ltheorie.

+ BOOLEAN cont_ltheorie(ltheorie,theorie) LTHEORIE ltheorie;
THEORIE theorie;

Theorie theorie contenue dans la liste de theories ltheorie ?

+ BOOLEAN egal_ltheorie(ltheorie1,ltheorie2) LTHEORIE ltheorie1,
ltheorie2;

Comparaison de deux listes de theories ltheorie1 et ltheorie2.

Conclusion.

Nous vivons dans un monde de machines et d'ordinateurs. Mais pour les faire fonctionner, il faut pouvoir communiquer avec elles. Bien des manières de communiquer de l'information existent. La parole nous est très naturelle et très pratique. Pourquoi donc ne pas l'utiliser pour communiquer avec les machines ? Cependant la mise en pratique de cette idée pose d'immenses problèmes qui ne sont aujourd'hui que partiellement résolus. Et il se peut même qu'on ne puisse jamais les résoudre complètement. De toute façon, si on n'essaie pas, on réalisera difficilement quelles sont les impasses.

On s'est donc attaqué progressivement aux différents problèmes. On a d'abord examiné les problèmes concernant la simple reconnaissance des différents éléments d'un signal acoustique. Et par la suite, on a continué en développant des systèmes plus perfectionnés pouvant 'comprendre' des phrases lorsque celles-ci respectaient tout un ensemble de contraintes (la compréhension de la parole peut être généralement définie comme le processus de transformation du signal acoustique continu en représentations discrètes auxquelles on peut associer une signification et qui, une fois comprises, peut être utilisée pour produire une réponse appropriée). Peu à peu, on a supprimé ces contraintes et trouvé des solutions aux problèmes qui apparaissaient du fait de la plus grande généralité du langage accepté. A ce jour, on se dirige vers des systèmes acceptant en entrée la parole continue et gérant un véritable dialogue avec l'interlocuteur dans le cadre d'une application particulière.

Ces nouveaux systèmes utilisent de nombreuses connaissances acoustiques et linguistiques et les progrès actuels se situent principalement du côté des connaissances sémantiques et pragmatiques. On fait d'ailleurs de plus en plus usage de connaissances relevant de l'intelligence artificielle. Cependant, il ne faut pas oublier que le niveau acoustique et les autres niveaux linguistiques laissent encore bon nombre de lacunes à

combler.

Le système de gestion de dialogues oraux finalisé de Nancy est destiné à une classe d'applications grand public de type " centre de renseignements ". Il a pour but d'examiner les différentes questions posées dans la gestion, par la machine, d'un véritable dialogue avec un interlocuteur. L'effort de recherche se situe principalement dans les domaines de la sémantique et de la pragmatique. Pour le reste, il se base pour beaucoup sur les acquis obtenus lors de la réalisation des systèmes précédents MYRTILLE 1 et 2 en les développant ou en les adaptant à ses besoins.

Personnellement, j'ai travaillé dans le domaine de l'analyse syntaxico-sémantique des énoncés avec l'aide de P. Mousel. On se base pour l'analyse syntaxique sur la grammaire développée par J-P. Pierrel, à savoir les Réseaux à Noeuds Procéduraux , et sur la grammaire de cas de G. Deville et H. Paulussen pour ce qui est de l'analyse sémantique. Le système MYRTILLE II a déjà permis de montrer la validité des R.N.P. pour l'analyse syntaxique mais, par contre, la grammaire de cas et ses règles permettant de transformer une représentation syntaxique fournie par les R.N.P. en une représentation sémantique obéissant aux règles de la grammaire de cas, n'a jamais été implémentée.

A l'intérieur du module SYN-SEM qui réalise ces analyses, se trouve un espace qui réalise le stockage des différentes solutions (les théories) qui apparaissent au cours de l'analyse d'un énoncé. Cet espace est appelé la base de théories. Le format des données qui composent une théorie dépend des analyseurs syntaxique et sémantique puisque ce sont les résultats de ces analyses qu'il s'agit de stocker. Mon travail visait à réaliser toute une série de fonctions d'accès vers cette base de théories pour permettre une utilisation plus simple de celle-ci. Malheureusement, nous nous trouvons dans un domaine qui est encore un domaine de recherche et où les connaissances évoluent régulièrement. Ceci a entraîné de nombreuses modifications dans la manière d'implémenter les analyseurs tout au long de l'année. Les structures de données traitées par les analyseurs ont donc été, de ce fait plus, d'une fois remaniées. Cela m'a alors obligé a recommencer plusieurs fois

le travail accompli, à reprendre la définition et la programmation des fonctions d'accès. Néanmoins, une grande partie du travail prévu à l'origine a été réalisée. Le temps qui m'était imparti et les derniers changements datant seulement de juillet 87 ne m'ont cependant pas permis d'atteindre pleinement le but fixé à l'origine. Il reste à réaliser la programmation des fonctions d'accès concernant les parties sémantiques des structures de données. Soulignons notre apport : en plus du fait qu'une grande partie du travail projeté a été réalisée, tous ces allers-retours nous ont permis de clarifier nos idées en ce qui concerne les analyses syntaxico-sémantiques par la machine et un grand nombre de choix ont pu être faits quant à l'implémentation finale. Un certain nombre de détails doivent encore être précisés.

Pour ce qui est de l'efficacité des fonctions d'accès, leur mise en service, lorsque les analyseurs seront terminés permettra d'évaluer la définition que nous en avons donnée.

page

Bibliographie.

{ Arte 84 }
Artefact, Micro-système, Septembre 84, pp 238-244.

{ Bruce 75 }
Bruce B. (1975)
" Case systems for natural languages ", Artificial Intelligence 6, pp 327-360.

{ Carb 85 }
Carbonnel N., Mangeol B., Mousel P., Pierrel J-M., Roussanaly A. (1985)
" Les sources de connaissance dans un système de dialogue oral homme-machine ", Actes du congrès ' Reconnaissance des formes et intelligence artificielle ', AFCET. Grenoble, France, Novembre 85.

{ Chom 71 }
Chomsky N. (1971)
" Aspects de la théorie syntaxique ", Editions du Seuil, Paris.

{ Deville 86 }
Deville G., Paulussen H. (1986)
" A case grammar as an original linguistic model for the semantic representation of utterances in a man-machine dialog system ", Memoire de linguistique informatique, Universitaire Instelling Antwerpen.

{ Falzon 85 }
Falzon P. (1985)
" Les langages opératifs ", Actes du séminaire GRECO-GALF ' Dialogue homme-machine à composante orale ', Nancy, France.

{ Fill 68 }
Fillmore C. (1968)
" The case for case ", dans Bach & Harms (eds.), Universals in Linguistic Theory, Holt Rinehart & Winston, New-york, pp 1-90.

{ Fohr 85 }
Fohr D., Carbonnell N., Haton J-P. (1985)
" SYSTEXP, un système expert pour le décodage acoustico-phonétique ". Actes des cinquièmes journées internationales sur les systèmes experts et leur applications. Avignon, France, Mai 1985.

{ Grévisse 69 }
Grévisse (1969)
" Le bon usage ", Duculot, Gembloux.

{ Harr 82 }
Harris Z. (1982)
" Discourse and sublanguage ", dans Kittredge & Lehrberger (eds), Sublangage, De Gruyter, Berlin/New-York, pp 231-236.

{ Haton 78 }
Haton J-P., Perennou G. (1978)
" Reconnaissance automatique de la parole ", dans 8ème Ecole d'été d'informatique de l'AFCET., Namur, Juillet 1978

{ Haton 84 }
Haton J-P. (1984)
" Compréhension de la parole et intelligence artificielle : l'état des recherches ", dans Proceedings of the seminar ' Man-Machine dialog by voice ', Nancy, October 1984.

{ Hill 71 }
Hill D.R. (1971)
" Automatic Speech Recognition : A Problem for Machine Intelligence ", dans Machine Intelligence 1+2, Collins Dale and Michie, Edinburgh University Press pp 199-226

{ Kitt 82a }
Kittredge R., Lehrberger J. (eds.) (1982)
" Sublanguage, studies of language in restricted semantic domains ", De Gruyter, Berlin/New-York.

{ Kitt 82b }
Kittedge R (1982)
" Variation and Homogeneity of sublangages ", dans Kittredge & Lehrberger (eds), Sublanguage, pp 107-137.

{ Lea 80 }
Lea W.A. (1980)
" Trends in speech recognition ", Prentice-Hall, Englewood Cliffs, New-Jersey.

{ Lehr 82 }
Lehrberger J. (1982)
" Automatic Translation and the Concept of Sublangage ", dans Kittredge & Lehrberger (eds), Sublangage, pp 81-106.

{ Lesser 75 }
Lesser V.R. et al. (1975)
" Organisation of the HEARSAY II Speech Understanding System ", IEEE trans, ASSP vol.23 n°1, 1975, pp 11-13.

{ Liber 70 }
Liberman A.M. (1970)
" The Grammar of Speech and Langage ", Cognitive psychology, n°1, 1970, 301-323.

{ Lorant 86 }
Lorant M., Simonet S. (1986)
" Contributions à un système de dialogue homme-machine à composante orale ", Mémoire de licence informatique, FNDP, Namur

{ Mariani 82 }
Mariani J. (1982)
" The ESOP Continuous Speech Understanding System. "
Proceedings of the IEEE-ICASSP.
Paris, France.

{ Mart 56 }
Martinet A. (1956)
" Eléments de linguistique générale ", Colin, Paris.

{ Meloni 83 }
Meloni H. (1983)
" Traitement des contraintes linguistiques en
reconnaissance automatique de la parole ", T.S.I., Vol.2,
N° 5, pp 349-363.

{ Mosk 82 }
Moskovitch W. (1982)
" What is a Sublanguage ? The Notion of Sublanguage in
Modern Soviet Linguistic ", dans Kittredge & Lehrberger
(eds.), Sublangage, pp 191-205.

{ Mousel 85 }
Mousel P. (1985)
" Le programme de création des réseaux à noeuds procéduraux
", Technical Report 85-R-092, CRIN, Nancy.

{ Mousel 87 }
Mousel P. (1987)
" La composante syntaxico-sémantique ", CRIN, Nancy.

{ Pierrel 81 }
Pierrel J-M. (1985)
" Contribution à la reconnaissance automatique du discours
continu ", Thèse de doctorat de spécialité, Université de
Nancy 1.

{ Pierrel 82 }
Pierrel J-M. (1982)
" Utilisation de contraintes linguistiques en compréhension
automatique de la parole continue : le système MYRTILLE II.
", T.S.I., pp 404-421.

{ Pierrel 85 }
Pierrel J-M. (1985)
" Aspect of man-machine voice dialog ", Actes du congrès
INRIA, Mai, Juin 1985, pp 253-278.

{ Schank 72 }
Schank R. (1972)
" Conceptual Dependency : a theory of a natural language
Understanding ", in Cognitive Psychology 3, pp 552-631.

{ Weizenbaum 84 }
Weizenbaum J. (1984)
" Computer Power and Human Reason, From Judgment to
Calculation ", London, Penguin Books.

{ Woods 73 }
Woods W. et Mackoul J. (1973)
" Mechanical inference Problems in Continuous Speech
Understanding ", dans Proceedings IJCAI Stamford 1973,
pp 200-207.

Annexe.

Sous-Réseaux R.N.P de MYRTILLE 2.

ANNEXE :

RNP subnetworks of MYRTILLE II (PIERREL 1981:356-363)

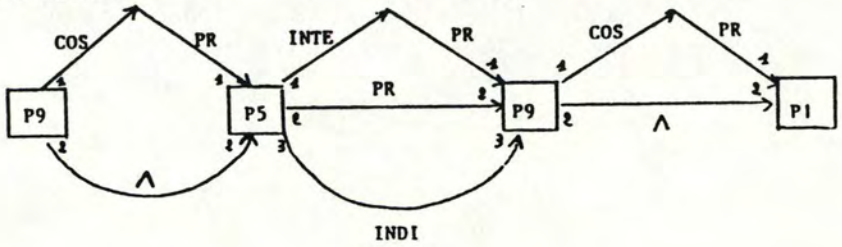
Liste des non terminaux (référence à un sous-réseau) :

CHIF	:	nombre
DATE	:	date
ENON	:	ensemble de l'énoncé
GC	:	groupe circonstanciel
GCOM	:	groupe complément
GL	:	groupe complément de lieu
GN	:	groupe nominal
GS	:	groupe sujet
GV	:	groupe verbal
INDI	:	interrogative indirecte
INF	:	infinitive
INTE	:	groupe interrogatif
PR	:	proposition
RELA	:	relative
SGV	:	suite verbale

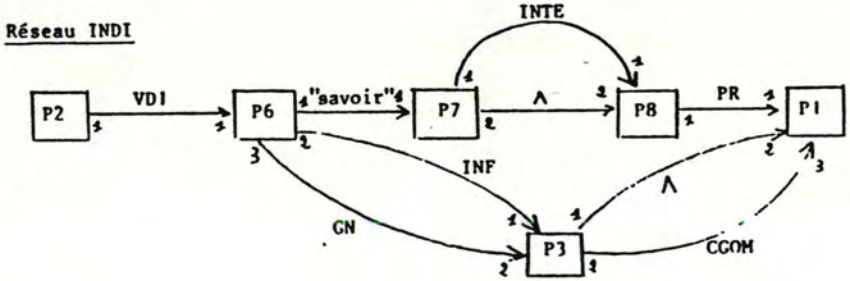
Liste des terminaux (accès au lexique) :

ADINT	:	adverbe interrogatif
ADJINT	:	adjectif interrogatif
ADJ	:	adjectif
ADL	:	adverbe de lieu
CØ	:	conjonction de coordination
COS	:	conjonction de subordination
DET	:	déterminant
DIZJ	:	10,20,30
DIZ1	:	20,30,40,50
DIZ2	:	60,80
GLP	:	groupe locution circonstancielle
JOUR	:	lundi,...,dimanche
MOIS	:	janvier,...,février
NOM	:	nom
NOMP	:	nom propre
PREP	:	préposition
PRINT	:	pronom interrogatif
PRL	:	préposition de lieu
PRN	:	pronom
PRSV	:	préposition dans SGV
UNIT1	:	1,...,9
UNIT2	:	1,...,19
VD1	:	je voudrais, j'aimerais, je désire(rais)...
V	:	verbe
^	:	élément vide

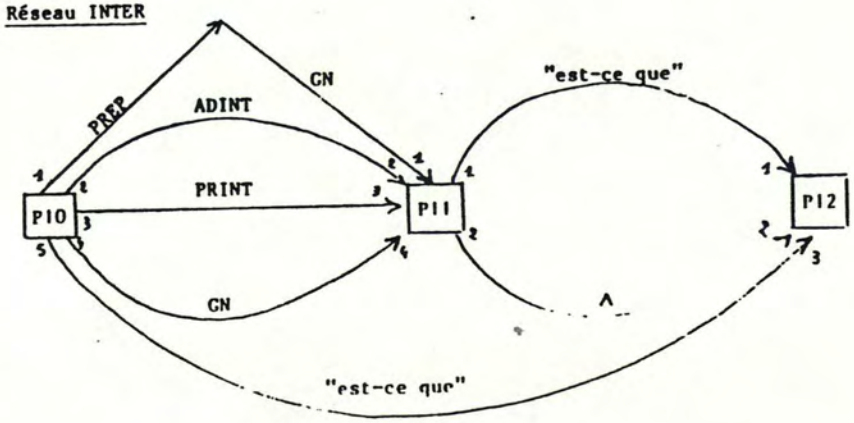
Réseau ENON



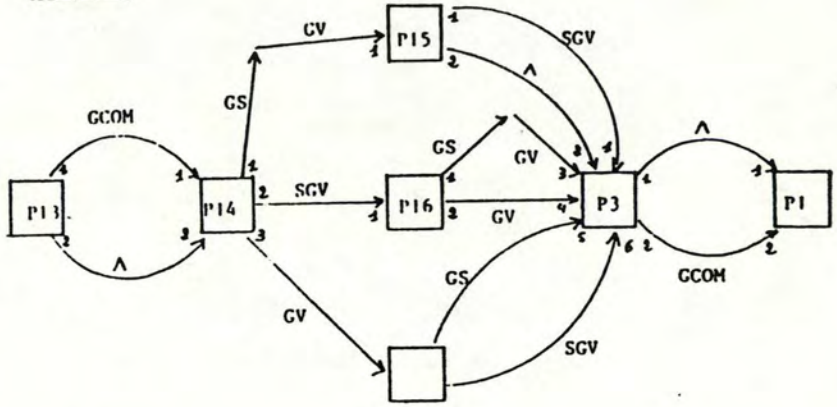
Réseau INDI



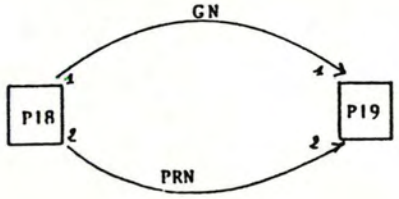
Réseau INTER



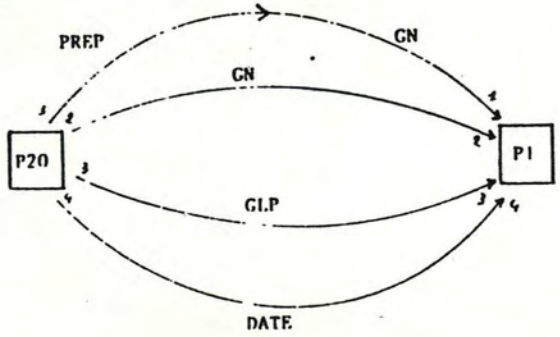
Réseau PR



Réseau GS

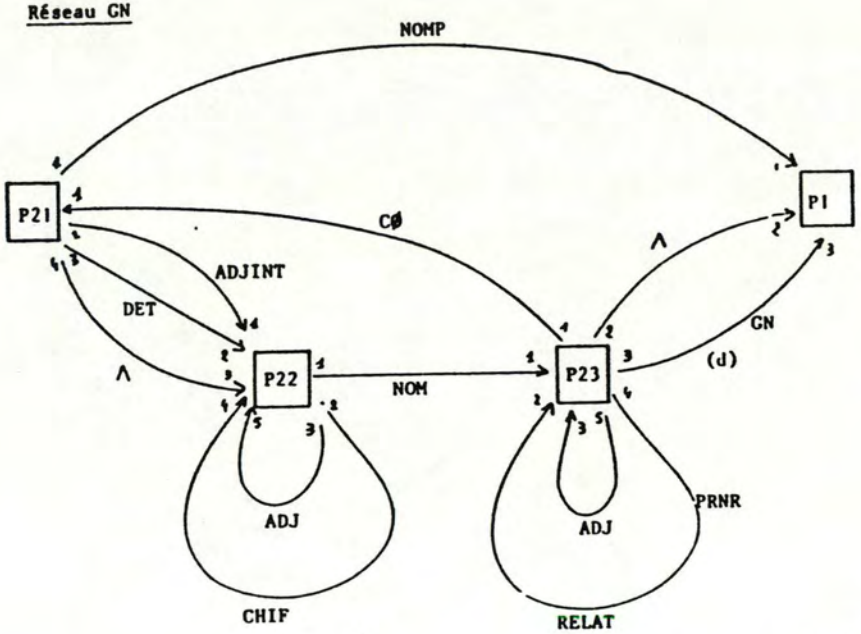


Réseau GC

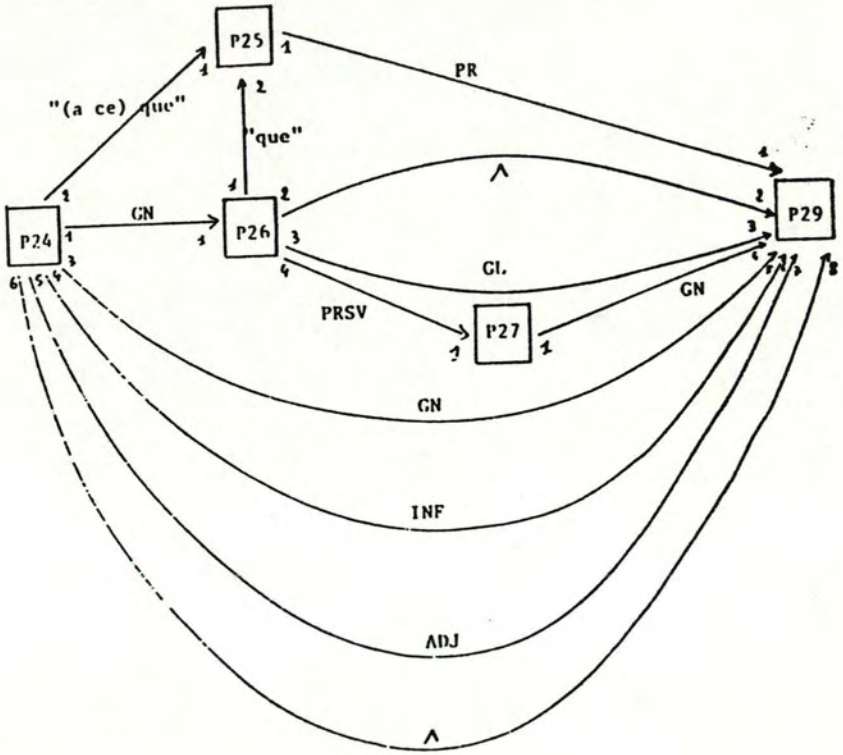


Annexe 2

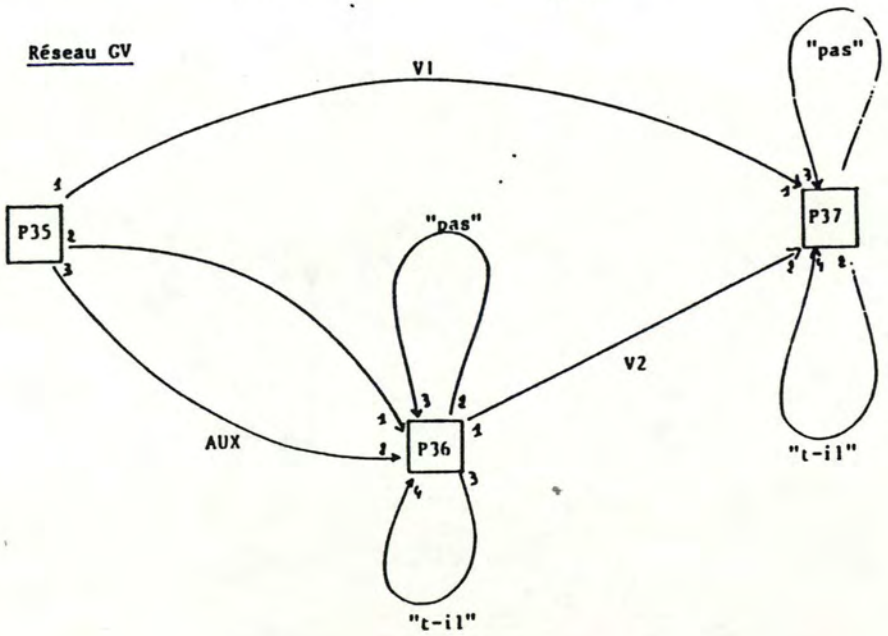
Réseau GN



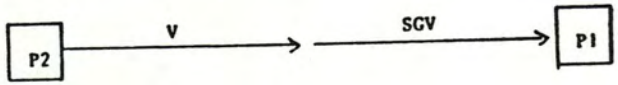
Réseau SGV



Réseau CV

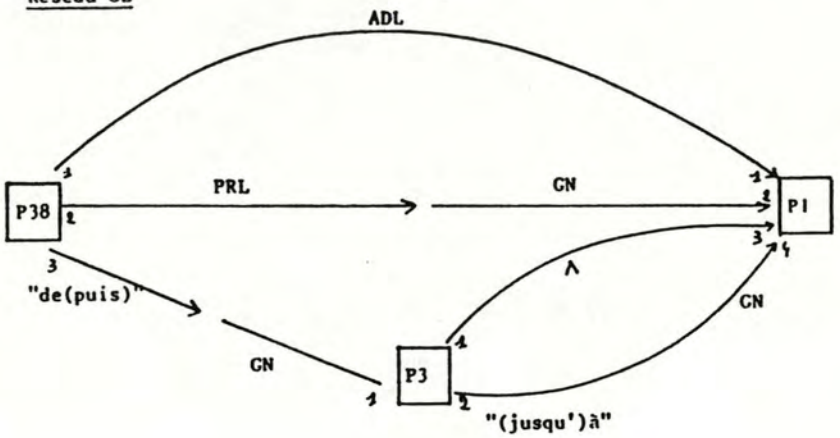


Réseau INF

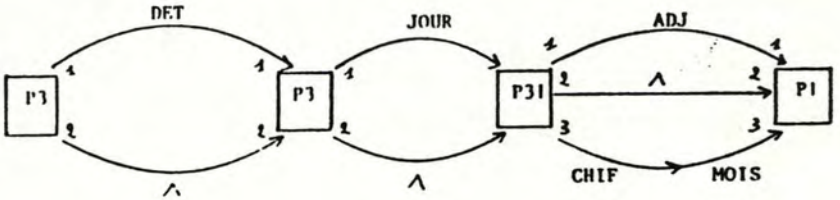


Annexe 3

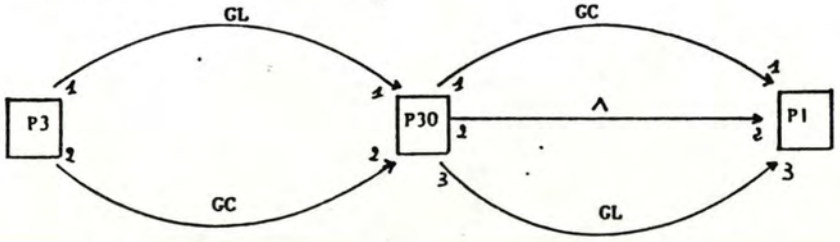
Réseau GL



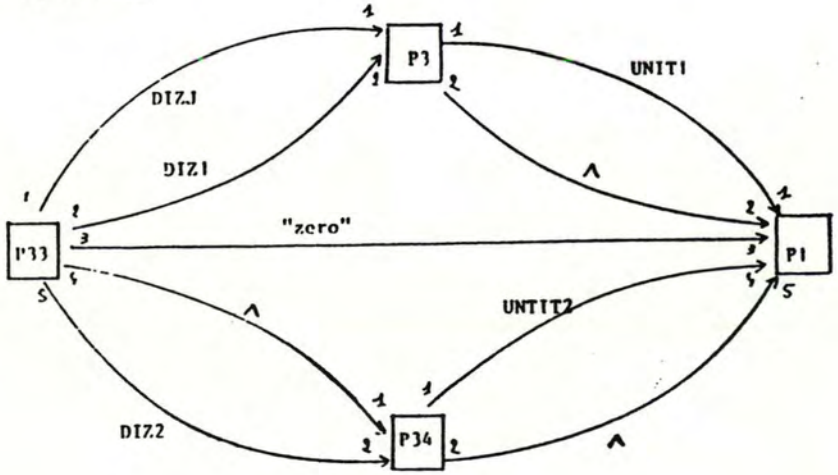
Réseau DATE



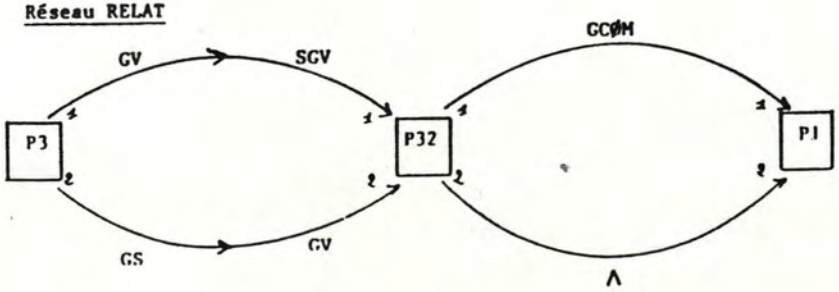
Réseau GCOM



Réseau CHIF



Réseau RELAT



Annexe 4